НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.89

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмний метод кластеризації веб-сайтів на основі аналізу пошукових запитів користувачів»

Виконала:
студентка ІІ курсу, групи КП-81мп
Білогуб Дар'я Сергіївна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент
Олещенко Любов Михайлівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри АУТС ФІОТ, к.т.н, доцент
Полторак Вадим Петрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (освітня програма) – 121 «Інженерія програмного забезпечення» ("Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем")

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студентці

Білогуб Дар'ї Сергіївні

1. Тема дисертації «Програмний метод кластеризації веб-сайтів на основі пошукових запитів користувачів», науковий керівник дисертації доцент кафедри ПЗКС, к.т.н., доцент Олещенко Любов Михайлівна, затверджені наказом по університету від «13» листопада 2019 р. № 3895-С.

2. Термін подання студентом дисертації «10» грудня 2019 р.

3. Об'єкт дослідження: процес аналізу ключових слів веб сторінок використовуючи інформацію про пошукові запити користувачів.

4. Предмет дослідження: моделі, методи, алгоритми кластерного аналізу пошукових запитів користувачів.

5. Перелік завдань, які потрібно розробити:
- оцінити сучасний стан проблеми, обґрунтувати актуальність напрямку досліджень, сформулювати мету та задачі дослідження;
- провести аналіз методів кластеризації;
- дослідити роботу існуючих методів кластеризації, які вирішують поставлену задачу дослідження;
- запропонувати та обґрунтувати шляхи модифікації обраного методу кластеризації для поставленої задачі;
- розробити модифікований метод кластеризації пошукових запитів користувачів на основі досліджених методів;
- виконати програмну реалізацію розробленого методу;
- виконати порівняння запропонованого методу із існуючими аналогами та надати оцінку отриманих результатів;
- описати бізнес-модель, що дозволить представити на ринку повноцінний програмний продукт із використанням представлених у роботі напрацювань.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
   - архітектура системи;
   - структура класів модифікованого алгоритму кластеризації;
   - результати роботи розробленого програмного забезпечення;
   - дерево проблем та рішень.

7. Орієнтовний перелік публікацій:
   - XII наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2019).

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., к.т.н., доцент		

9. Дата видачі завдання «25» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.11.2018	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2019	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2019	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2019	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2019	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2019	13.11.2019	
8.	Оформлення текстової і графічної частини магістерської дисертації	05.12.2019	

Студент Д.С. Білогуб

Науковий керівник дисертації Л.М. Олещенко

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Семантичне ядро – упорядкований набір пошукових слів, їх морфологічних форм і словосполучень, які найбільш точно характеризують вид діяльності, товар або послугу, пропоновані сайтом [1].

Кластерний аналіз – задача розбиття заданої вибірки об'єктів (ситуацій) на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися [2].

Кластер – група однакових або подібних елементів, зібраних разом або близько один до одного.

k-means – популярний метод кластеризації, – впорядкування множини об'єктів в порівняно однорідні групи [3].

Граф – це сукупність об'єктів із зв'язками між ними.

Вага – значення, поставлено у відповідність даному ребру зваженого графа. Зазвичай вага – дійсне число, в такому випадку його можна інтерпретувати як «довжину» ребра [4].

ВСТУП

З огляду на постійну боротьбу пошукових систем з різними накрутками посилальних чинників, правильна структура сайту все більше виходить на перший план при проведенні пошукової оптимізації сайту. Один з основних ключів для грамотної опрацювання структури сайту – є максимально детальне опрацювання семантичного ядра.

Семантичне ядро відповідає на глобальне питання: яку інформацію можна знайти на сайті. Оскільки одним з головних принципів бізнесу і маркетингу вважається клієнтоорієнтованість, на створення семантичного ядра можна дивитися з різних боків. Спершу, необхідно визначити, за допомогою яких пошукових запитів користувачі шукають інформацію, яка буде опублікована на сайті. Також, побудова смислового ядра вирішує ще одну задачу. Йдеться про розподіл пошукових фраз по сторінках ресурсу. Працюючи з ядром, необхідно коректно визначити яка сторінка найточніше відповідає на конкретний пошуковий запит або групу запитів.

Аналізуючи характеристики кластерів користувачів, можна краще зрозуміти користувачів Інтернету, і, таким чином, можна отримати більш релевантні веб-документи. Для ідентифікації підгруп користувачів використовується кластеризація, щоб класифікувати користувачів на основі їх поведінки, а саме їх пошукових запитів. Кожен метод кластеризації використовує різні критерії для групування об'єктів даних. Групування клієнтів або елементів даних, що мають подібні характеристики, дає найкращі результати.

Проблема кластеризації (або сегментації) пошукових запитів веб-користувачів полягає у розділенні наборів запитів на кластери так, щоб запити в одному кластері були більш подібними між собою, ніж в інших кластерах.

Аналіз наявних робіт показав, що на даний час існує досить багато методів кластеризації веб-користувачів. Однак, безпосереднє застосування цих методів з простими наборами даних не є достатньо ефективним і може відображати цікаві кластери для дослідження, оскільки веб-сервер зазвичай

містить тисячі і навіть мільйони сторінок, а користувачі Інтернету можуть отримувати доступ до веб-сторінок з різноманітними цілями.

Існуючі програмні засоби, що використовуються для вирішення поставленої задачі, мають ряд недоліків. Вони використовують алгоритми семантичної кластеризації для розпізнавання тематично пов'язаних веб-сторінок. Більшість із них покладається на аналіз тексту вмісту веб-документів, і це призводить до певних обмежень, таких як тривалий час обробки, потреба в представницькому текстовому вмісті чи розмитості природних мов. Основною проблемою при використанні даних алгоритмів є вибір числа кластерів, що зазвичай обирається навмання.

Тому відповідно до мети написання дипломного проекту поставлені і розв’язані наступні задачі:

– огляд та проведення аналізу існуючих аналогів, зокрема програмних рішень та відомих технологій для виділення їх переваг та недоліків, на основі яких відбувається виявлення вимог для розроблення програми;
– розробка модифікованого методу кластеризації, на основі проведеного аналізу;
– вибір базових технологій для реалізації модифікованого методу кластеризації;
– реалізація й тестування програмного забезпечення та проведення аналізу отриманого результату.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ КЛАСТЕРИЗАЦІЇ ПОШУКОВИХ ЗАПИТІВ

1.1. Аналіз основних методів кластеризації

Кластеризація – це технологія машинного навчання, яка включає групування точок даних. Враховуючи набір точок даних, ми можемо використовувати алгоритм кластеризації для класифікації кожної точки даних у конкретній групі. Теоретично точки даних, що знаходяться в одній групі, повинні мати подібні властивості та особливості, тоді як точки даних у різних групах повинні мати сильно відрізняються властивості та особливості. Кластеризація – це метод непідвладного навчання і є загальною методикою статистичного аналізу даних, що використовується в багатьох галузях.

Задача кластеризації відноситься до широкого класу задач навчання без учителя. Таким чином, необхідно розбити вибірку об'єктів на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися.

1.1.1. *Алгоритми теорії графів*

Кластеризація графів відноситься до кластеризації даних у вигляді графів. На графічних даних можна виконати дві чіткі форми кластеризації. Вершинна кластеризація намагається об'єднати вузли графа в групи щільно з'єднаних областей на основі або крайових ваг, або реберних відстаней. Друга форма кластеризації графів трактує графіки як об'єкти, що підлягають кластеризації, та виконує кластеризацію цих об’єктів на основі їх подібності. Другий підхід часто зустрічається в контексті структурованих або *XML*-даних.

Дані для кластеризації можуть бути представлені у вигляді графа, де кожен елемент, який повинен бути кластеризований, представлений у вигляді вузла, а відстань між двома елементами моделюється певною вагою на межі, що з'єднує вузли [6]. Таким чином, в кластеризації графів елементи всередині кластера з'єднані один з одним, але не мають зв'язку з елементами поза цим кластером. Крім того, деякі нещодавно запропоновані підходи виконують

кластеризацію безпосередньо на основі графів. Важливим підходом до кластерів на основі графів – це кластери, засновані на суміжності інформації в реченні.

Суть таких алгоритмів полягає в тому, що вибірка об'єктів представляється у вигляді графа $G = (V, E)$. На рис. 1 зображено візуалізацію кластеризації з використанням теорії графів. Головна особливість кластеризації графів – це відсутність відстані між двома довільними точками в просторі, тому що немає самого простору, немає норми, і неможливо визначити відстань. Замість цього, є метадані ребер (в ідеалі, для кожної пари вершин). Якщо є «вага» ребра, то його можна інтерпретувати як відстань (або, навпаки, як схожість), і тоді визначені відстані для кожної пари вершин.

Багато з алгоритмів кластеризації в евклідовому просторі підходять і для графів, так як для цих алгоритмів потрібно лише знати відстань між спостереженнями, а не між довільними «точками в просторі». У графів є багато своїх унікальних властивостей, які теж можна використовувати: компоненти зв'язності, локальні скупчення ребер, місця зациклення інформаційних потоків та ін.

Вершинам графа відповідають об'єкти вибірки, а ребрам – попарні відстані між об'єктами $\rho_{ij} = (x_i, x_j)$.

Перевагою алгоритмів кластеризації на основі графів є наочність, відносна простота реалізації і можливість різних удосконалень, заснованих на геометричних міркуваннях. Основними алгоритмами є алгоритм виділення зв'язкових компонент, алгоритм побудови мінімального кістякового дерева і алгоритм пошарової кластеризації.

Розглянемо алгоритм виділення зв'язкових компонент. Задається параметр R і в графі видаляються всі ребра (i, j), для яких $\rho_{ij} > R$.

Сполученими залишаються тільки найбільш близькі пари об'єктів. Ідея даного алгоритму полягає в тому, щоб підібрати таке значення параметру $R \in [\min \rho_{ij}, \max \rho_{ij}]$, при якому граф розвалиться на декілька зв'язкових компонент. Знайдені зв'язкові компоненти – i є кластери.

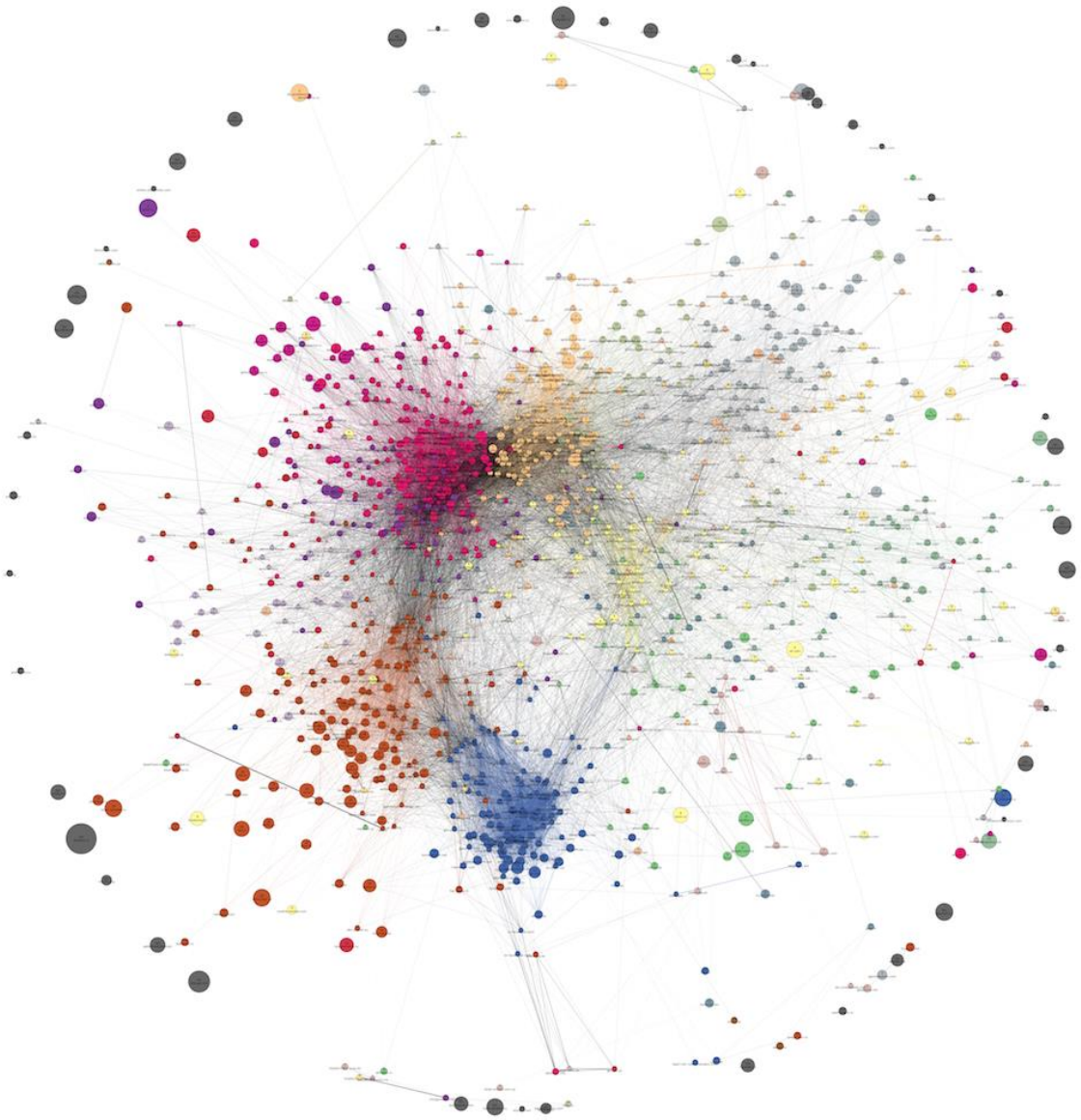


Рис. 1. Візуалізація графа

Серед недоліків даного алгоритму можна виділити наступні:

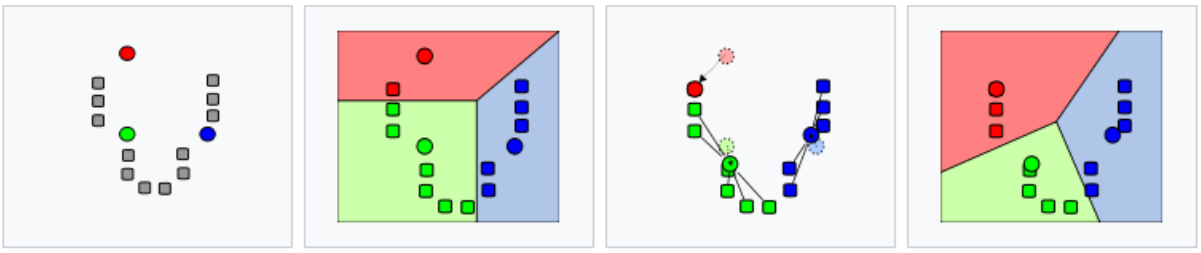
- обмежена застосовність. Алгоритм виділення зв'язкових компонент найбільш підходить для виділення кластерів типу згущень або стрічок. Наявність розрідженого фону або «вузьких перемичок» між кластерами призводить до неадекватної кластеризації;
- погана керованість числом кластерів. Для багатьох додатків зручніше задавати не параметр *R*, а число кластерів або деякий поріг «чіткості

кластеризації». Управляти числом кластерів за допомогою параметра R досить важко. Доводиться кілька разів вирішувати завдання при різних R, що негативно позначається на тимчасових витратах.

1.1.2. *Статистичні алгоритми кластеризації*

До статистичих алгоритмів кластеризації належать такі алгоритми, як *k*-means та *EM*-алгоритм.

Метод *k*-середніх – найбільш популярний метод кластеризації. Алгоритм є версією *EM*-алгоритму. Він розбиває безліч елементів векторного простору на заздалегідь відоме число кластерів k [7].



1. *k* початкових «середніх» (тут *k*=3) випадково згенеровані у межах домени даних (кольорові).

2. створено *k* кластерів, асоціюючи кожне спостереження з найближчим середнім. Розбиття відбувається згідно з діаграмою Вороного утвореною середніми.

3. Центроїд кожного з *k* кластерів стає новим середнім.

4. Кроки 2 і 3 повторюються до досягнення збіжності.

Рис. 2. Демонстрація алгоритму *k*-means

Основна ідея полягає в тому, що на кожній ітерації переобчислюють центр мас для кожного кластера, отриманого на попередньому кроці, потім вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився ближчим за обраною метрикою, як показано на рис. 2.

Основними недоліками алгоритму є:

– не гарантується досягнення глобального мінімуму сумарного квадратичного відхилення V, а тільки одного з локальних мінімумів;
– результат залежить від вибору вихідних центрів кластерів, їх оптимальний вибір невідомий;
– число кластерів треба знати заздалегідь.

EM-алгоритм – алгоритм, який використовується в математичній статистиці для знаходження оцінок максимальної правдоподібності параметрів імовірнісних моделей, в разі, коли модель залежить від деяких прихованих змінних. Кожна ітерація алгоритму складається з двох кроків. На *E*-кроці (expectation) обчислюється очікуване значення функції правдоподібності, при цьому приховані змінні розглядаються як спостережувані. На *M*-кроці (maximization) обчислюється оцінка максимальної правдоподібності, таким чином збільшується очікуване значення правдоподібності, що обчислюється на *E*-кроці. Потім це значення використовується для *E*-кроку на наступній ітерації. Алгоритм виконується до збіжності.

Ідея алгоритму полягає в наступному. Штучно вводиться допоміжний вектор прихованих змінних G, що володіє двома властивостями. З одного боку, він може бути обчислений, якщо відомі значення вектора параметрів Θ. З іншого боку, пошук максимуму правдоподібності сильно спрощується, якщо відомі значення прихованих змінних [8].

EM-алгоритм складається з ітераційного повторення двох кроків:

1. На *E*-кроці обчислюється очікуване значення (expectation) вектора прихованих змінних за поточним наближенню вектора параметрів.
2. На *M*-кроці вирішується завдання максимізації правдоподібності (maximization) і знаходиться наступне наближення вектора параметрів за поточними значеннями векторів прихованих змінних і параметрів. Таким чином, *M*-крок зводиться до обчислення ваг компонент w_j як середніх арифметичних і оцінювання параметрів компонент Θ_j шляхом вирішення k незалежних оптимізаційних задач. Відзначимо, що поділ змінних виявилося можливим завдяки вдалому введенню прихованих змінних.

Серед недоліків можна виділити наступні:

– основний алгоритм нестійкий за початковими даними (тобто тим, які ініціалізують вектор параметрів на першій ітерації), так як він знаходить локальний екстремум, значення якого може виявитися набагато нижче, ніж глобальний максимум. Залежно від вибору початкового наближення алгоритм може сходитися до різних точок. Також може сильно варіюватися швидкість збіжності;
– основний алгоритм не дозволяє визначати кількість k компонент суміші. Ця величина є структурним параметром алгоритму.

1.1.3. *Ієрархічна кластеризація*

Ієрархічна кластеризація – найпопулярніший і широко застосовуваний метод аналізу даних, який краще всього застосовувати на невеликих наборах даних. У цьому методі вузли порівнюють один з одним за ознакою подібності. Більші групи будуються приєднанням до груп вузлів виходячи з їх подібності. Вводиться критерій для порівняння вузлів на основі їх взаємозв'язку, який в результаті приводить до структури кластеризації, що складається з вкладених розділів. Стандартний алгоритм ієрархічної кластеризації має часову складність $\mathcal{O}(n^2)$ та потребує $\mathcal{O}(n^2)$ пам'яті, що занадто повільно для наборів даних середнього розміру.

Ієрархічна кластеризація – сукупність алгоритмів упорядкування даних, спрямованих на створення ієрархії (дерева) вкладених кластерів. Ієрархічні алгоритми кластеризації – це класичні алгоритми кластеризації, де створюються набори кластерів. В ієрархічних алгоритмах в якості введення використовується матриця суміжності вершин $n \times n$, а матриця суміжності містить значення відстані, а не просте булеве значення [9]. Стратегії побудови ієрархічної кластеризації діляться на два типи:

– агломератові (об'єднувальні). Це підхід «знизу-вгору». Спочатку кожна точка має власний кластер, а далі пари кластерів об'єднуються при підйомі по ієрархії;

– розділювальні. Це підхід «згори-вниз». Спочатку всі точки знаходяться у єдиному кластері, потім відбувається рекурсивне розбиття при русі вниз по ієрархії.

Алгоритми ієрархічної кластеризації припускають, що множина об'єктів, яка аналізується характеризується певним ступенем зв'язності [9]. За кількістю ознак іноді виділяють монотетичні та політетичні методи класифікації. Зазвичай розбиття та об'єднання визначаються у жадібний спосіб. Отриману ієрархію типово зображають як дендрограму, я показано на рис. 3.

Під дендрограмою зазвичай розуміється дерево, побудоване по матриці заходів близькості. Дендрограма дозволяє зобразити взаємні зв'язки між об'єктами з заданої множини. Для створення дендрограми потрібно матриця подібності (або відмінності), яка визначає рівень подібності між парами кластерів. Найчастіше використовуються агломеративні методи.

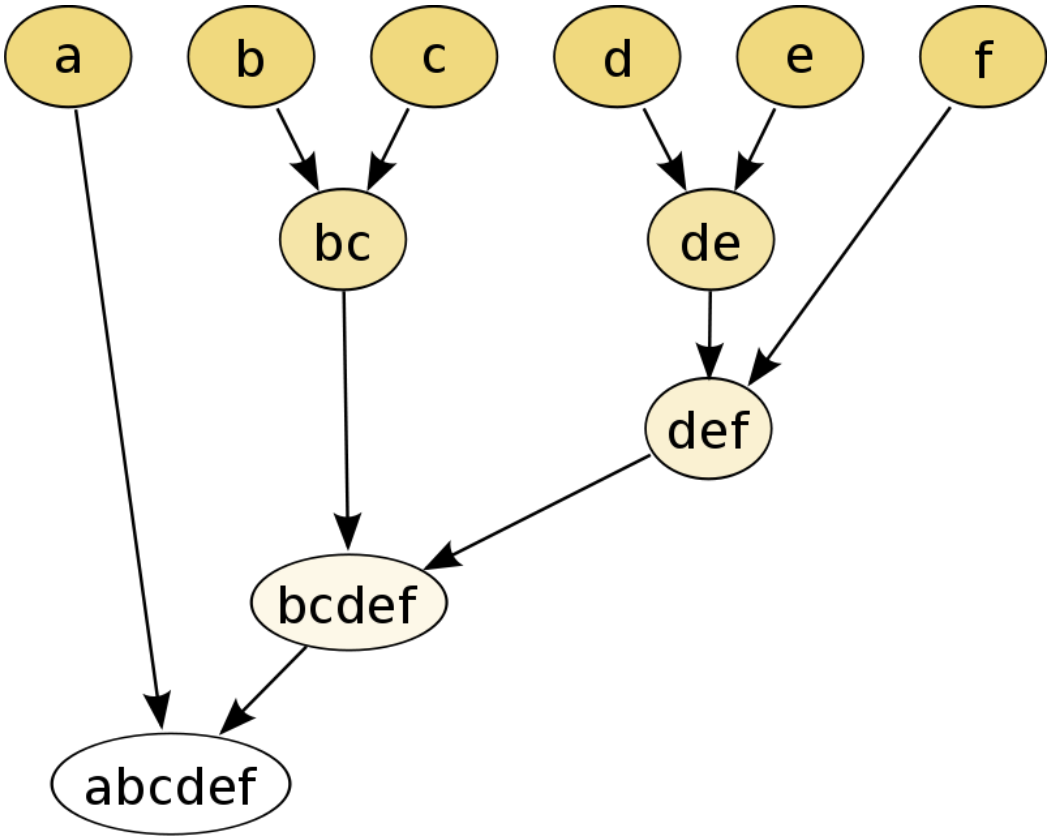


Рис. 3. Дендограма

Для побудови матриці подібності (відмінності) необхідно задати міру відстані між двома кластерами. Найбільш часто використовуються наступні методи визначення відстані:

- метод одиночного зв'язку, також відомий як «метод найближчого сусіда»;
- метод повного зв'язку, також відомий як «метод далекого сусіда»;
- метод середнього зв'язку;
- центроїдний метод;
- метод Уорда.

1.1.4. *Нейронні мережі Кохонена*

Нейронні мережі Кохонена – клас нейронних мереж, основним елементом яких є шар Кохонена. Шар Кохонена складається з адаптивних лінійних суматорів («лінійних формальних нейронів»). Як правило, вихідні сигнали шару Кохонена обробляються за правилом «Переможець отримує все»: найбільший сигнал перетворюється в одиничний, інші звертаються в нуль.

За способами налаштування вхідних ваг суматорів і по вирішенню задач розрізняють багато різновидів мереж Кохонена. Найбільш відомі з них:

- мережі векторного квантування сигналів, тісно пов'язані з найпростішим базовим алгоритмом кластерного аналізу (метод динамічних ядер або *k*-середніх);
- самоорганізовані карти Кохонена (англ. self-organising maps, SOM);
- мережі векторного квантування, яких навчають з учителем (англ. learning vector quantization).

Шар Кохонена складається з n паралельно діючих лінійних елементів. Всі вони мають однакову кількість входів m і отримують на свої входи один і той же вектор вхідних сигналів $x = (x_1, \dots x_m)\}$. На виході j-го лінійного

елемента отримуємо сигнал: $y_j = w_{j0} + \sum_{i=1}^{m} w_{ji} x_i$, де w_{ij} – ваговий коефіцієнт i-го входу j нейрона, w_{i0} – пороговий коефіцієнт.

Після проходження шару лінійних елементів сигнали посилаються на обробку за правилом «переможець забирає все»: серед вихідних сигналів y_j шукається максимальний; його номер $j_{max} = \arg\max\{y_i\}$. Остаточно, на виході сигнал з номером j_{max} дорівнює одиниці, решта – нулю. Якщо максимум одночасно досягається для декількох j_{max}, то або приймають всі відповідні сигнали рівними одиниці, або тільки перший у списку (за згодою).

1.2. Огляд існуючих комерційних програмних продуктів

1.2.1. *Datawiz.io*

Компанія Datawiz.io надає онлайн-сервіси для аналітики даних в рітейлі і ресторанному бізнесі. Основною задачею компанії – є автоматизація процесів аналізу: чеків, продаж і даних програми лояльності.

Сервіс допомагає середнім і малим продуктовим мережам:

- отримати інформацію про покупців;
- проаналізувати показники продажів по мережі і кожному магазину;
- визначити ключові товари, пари товарів;
- оптимальну ціну товару;
- спланувати промо-акції для збільшення прибутку.

Функціонал BI Datawiz.io дозволяє формувати аналітику в розрізі магазинів, товарних груп і товарів для оптимізації асортименту, управління ціновою політикою мережі, планування маркетингової активності та визначення типових моделей поведінки клієнтів. Datawiz.io використовує кластеризацію як метод групування клієнтів за даними про їх поведінку – покупки, банківські транзакції, кредитні історії.

Для кластеризації масиву даних (чеки, дані по програмах лояльності) використовується алгоритм *k*-means. Він добре масштабується і оптимізується під Hadoop. Також, в якості альтернативи використовується алгоритм Affinity Propagation. У нього є ряд істотних мінусів: він повільний і погано

масштабується. Але в окремих випадках, при бажанні і наявності вільного часу, можна використовувати його для кластеризації на коротких проміжках часу.

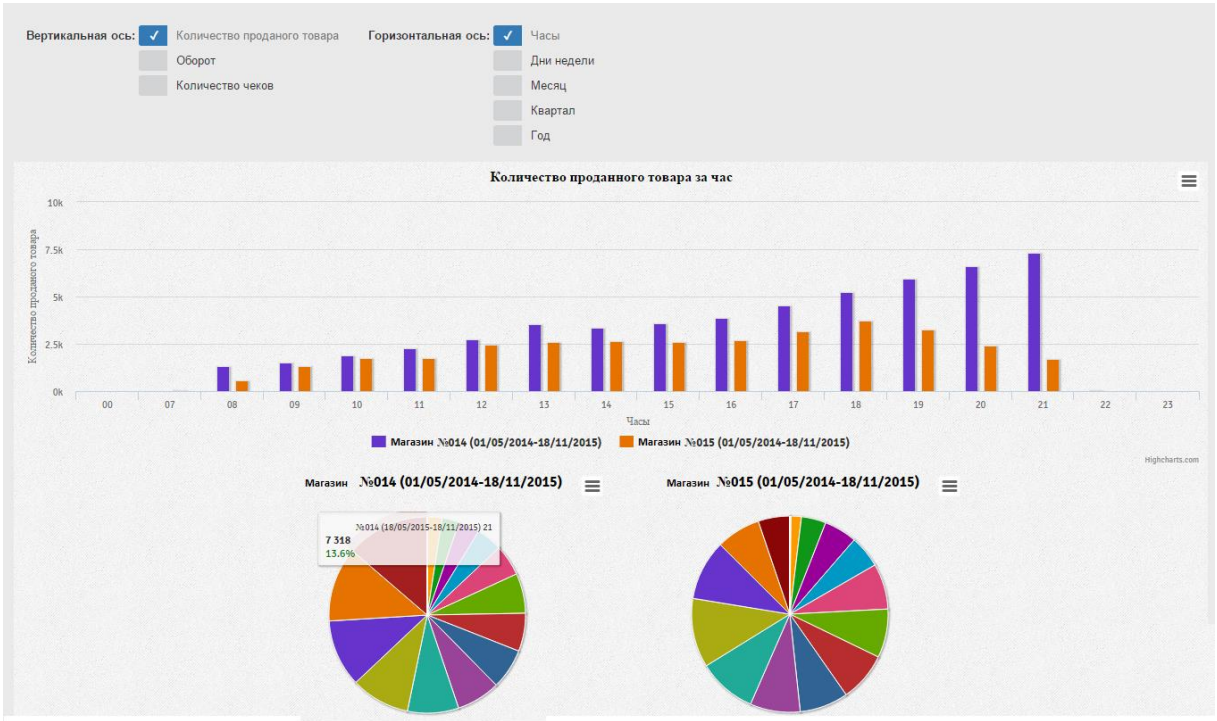


Рис. 4. Приклад роботи додатку Datawiz.io

Як згадувалось вище, алгоритм *k*-means потребує початкового визначення кількості кластерів. Отже, в даному додатку кількість кластерів обирається експериментальним шляхом, виходячи з власного досвіду. Мала кількість кластерів буде малоефективно і не інформативно, тому що в такому випадку буде отримано один-два «мегакластер», куди буде входити 98% клієнтів і кілька непотрібних маленьких кластерів.

При великій кількості кластерів вийде занадто багато маленьких груп. До того ж ніхто не хоче аналізувати 5000 окремих дрібних кластерів. Для кожного окремого випадку повинен бути свій індивідуальний підхід. Для тривалих періодів і великої кількості кластерів використовується *k*-means.

1.2.2. *Data-Centric Alliance*

DCA – російська компанія, створює технології цифрового маркетингу і розробляє продукти на базі Big Data і Programmatic. Має у своєму розпорядженні один з найбільш об'ємних масивів анонімних даних про користувачів російського інтернету. DCA допомагає маркетологам і аналітикам дізнаватися про переваги і поведінку людей в інтернеті і реальному житті. Займається розміщенням реклами на RTB-аукціонах і просуванням в соціальних мережах. Має підрозділ R&D для розробки нових інструментів, які дозволяють більше дізнаватися про користувачів інтернету і ефективніше розміщувати рекламу.

Для свого додатку компанія виконує кластеризацію графів, що були утворені за допомогою алгоритму *k*-medoids, розроблений мовою Python. У результататі спершу відбувається – кластеризація веб-доменів. Використовуючи базу DMP (дані про відвідування користувачами різних доменів), будується граф, де в якості вузлів виступають домени, а в якості ребер – афініті між доменами. Афініті (він же ліфт) між доменами x і y – це вибіркова оцінка того, наскільки події «відвідування користувачем u домену x» і «відвідування користувачем u домену y» близькі до незалежності.

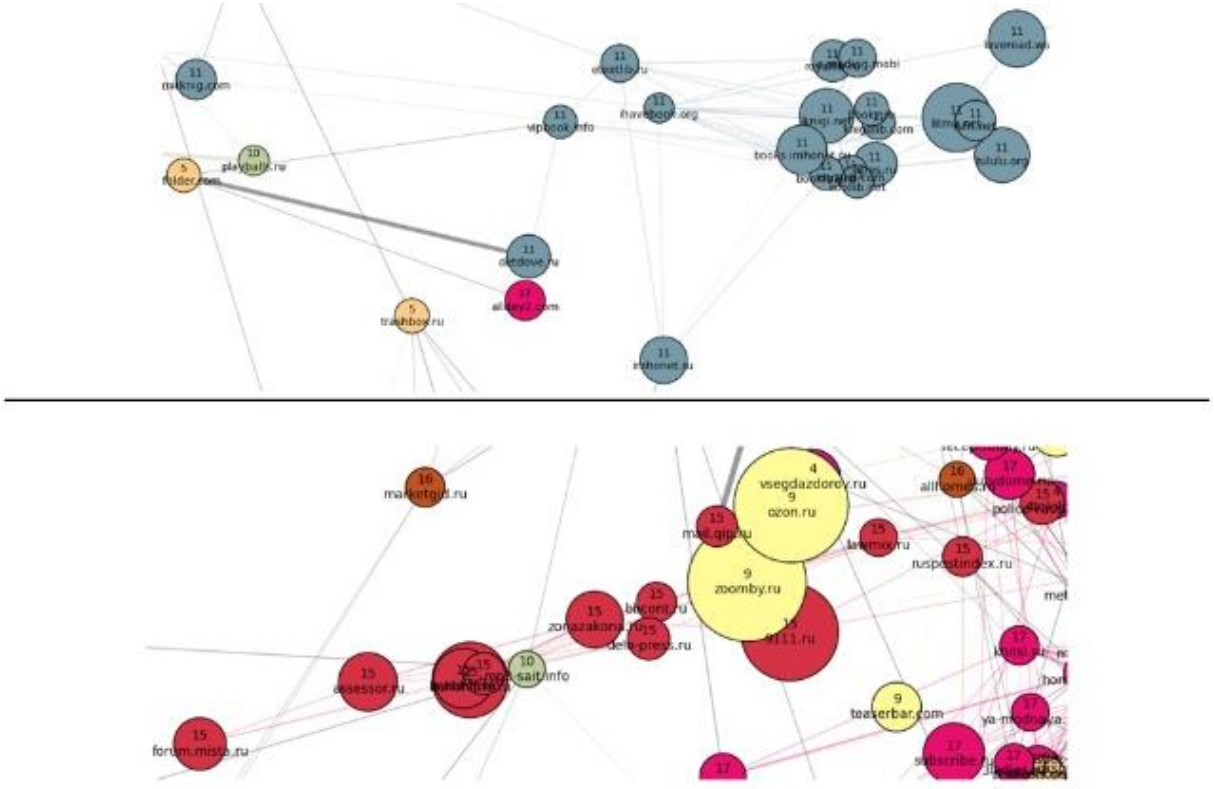


Рис. 5. Приклад роботи додатку

1.3. Висновки до розділу 1

У даному розділі було проведено аналіз існуючих методів кластеризації та аналогів до поставленої задачі. Отримана інформація була систематизована з метою використання в подальшій роботі для визначення вимог до програмної розробки, що дозволить створити додаток для кластеризації пошукових запитів користувачів.

Існує безліч практичних застосувань кластеризації як в сфері інформаційних технологій, так і в інших областях. Найпоширеніші приклади застосування кластеризації:

1. Аналіз даних:
   - спрощення роботи з інформацією;
   - візуалізація даних.
2. Витяг і пошук інформації:
   - побудова зручних класифікаторів.
3. Угруповання і розпізнавання об'єктів:

- розпізнавання образів;
- угруповання об'єктів.

Кластерний аналіз можна представити у вигляді такої послідовності дій:

1. Вибір безлічі об'єктів.
2. Визначення безлічі змінних, для оцінки об'єктів і складання векторів характеристик.
3. Нормування векторів характеристик одним з доступних методів.
4. Визначення подібності між об'єктами по заданій метриці.
5. Застосування обраного методу кластерного аналізу для розбиття безлічі об'єктів на кластери по ступеню схожості.
6. Представлення результатів аналізу.

На основі дослідження методів кластеризації та їх порівняння було сформовано мету дослідження та вимоги до розроблюваного в даній роботі програмного продукту.

Метою даної роботи є оптимізація релевантності документа запиту користувача шляхом розробки програмного забезпечення з використанням методів кластеризації пошукових запитів користувачів.

Виклад рішення наведено у розділі 2 цієї дисертації.

2. МОДИФІКОВАЦІЯ МЕТОДУ КЛАСТЕРИЗАЦІЇ ПОШУКОВИХ ЗАПИТІВ КОРИСТУВАЧІВ

Рішення, що пропонується в даній дисертації, сформувалося в результаті поступового вирішення проблем. Тому даний розділ описуватиме проблеми, задачі та їх вирішення у тому порядку, як вони виникали в процесі роботи.

Виходячи з того, що основною метою цієї роботи є розробка такого програмного додатку, що виконуватиме кластеризацію пошукових запитів користувачів з більшою точністю.

2.1. Аргументація вибору базового методу кластеризації даних

В результаті розгляду методів існуючих методів кластеризації, які можна застосувати для вирішення поставленої задачі можна виділити на 3 головні категорії:

- графові алгоритми кластеризації;
- статистичні алгоритми кластеризації;
- ієрархічні алгоритми кластеризації.

За алгоритмічною основою на:

- адаптивні та неадаптивні;
- глобальні та децентралізовані;
- статичні та динамічні.

За базовими вимогами:

- стабільності;
- оптимальності;
- точності;
- простоти;
- надійності;
- продуктивності.

Дані характеристики показують список важелів, що можуть вплинути на розробку покращень нового алгоритму.

В рамках розробки та в рамках підтримки режиму швидкого розгортання, необхідно базуватися на динамічних алгоритмах. Динамічні алгоритми в свою чергу забезпечують сумісність з динамічними мережами, та зміні стану в залежності від конфігурацій мережі, та від зовнішнього впливу на мережу, за час функціонування графу та взаємодії вузлів мережі.

Також за базовими вимогами необхідно обрати алгоритм з найбільшою продуктивністю та одночасно з якомога більшим кількістю інших характеристик, що не будуть конфліктувати між собою, наприклад, мати в собі продуктивність, цей стан вимагає достатньої кількості проміжних валідацій, що можуть вимагати додаткового процесорного часу на обробку таких ситуацій, що в кінцевому випадку значно знизить продуктивність самого алгоритму.

Під даний опис підпадають декілька алгоритмів, а саме *k*-means та алгоритми основані на теорії графів – алгоритми, що схожі за характеристиками, але різні за своєю природою.

2.2. Модифікація обраного методу кластеризації даних

В даному розділі буде розглянуто модифікований метод кластеризації, коли дані, що підлягають кластеризації, будуть представлені графами замість векторів чи інших моделей. Зокрема, буде розширено класичний алгоритм кластеризації *k*-means для роботи з графами, що представляють веб-документи. Традиційна парадигма в цих методах полягає в трактуванні проблеми кластеризації як графік: вузли представляють елементи, що підлягають кластеризації, а ваги на краях, що з'єднують два вузли, вказують на відстань (несхожість) між об'єктами, які представляють вузли. Після застосування алгоритму пов’язані компоненти вказують, до яких об’єктів належать кластери: об’єкти, вузли яких з'єднані ребрами, знаходяться в одному кластері.

Алгоритм кластеризації *k*-means – один з найпопулярніших і найпростіших методів кластеризації даних. Звичайним способом є подання

елементів даних у вигляді сукупності n-числових значень, зазвичай розташованих у векторній формі в просторі R^n. Тоді евклідова відстань у цьому просторі та центроїд набору векторів використовуються для обчислення середнього значення даних у кластері. Основний алгоритм реалізації k-means наведено у табл. 1.

Таблиця 1

Традиційний алгоритм кластеризації k-means

Вхідні дані:	набір n елементів даних та параметр k, що визначають кількість кластерів для створення
Вихідні дані:	центроїди кластерів і кластер (ціле число в межах $[1, k]$), для кожного елемента даних якому він належить
Крок 1.	Призначити кожен елемент даних випадковим чином кластеру (від 1 до k).
Крок 2.	Використовуючи початкове призначення, визначити центроїди кожного кластеру.
Крок 3.	З огляду на нові центроїди, призначити кожний елемент даних так, щоб він знаходився в кластері його найближчого центроїда.
Крок 4.	Повторно обчислити центроїди, як на кроці 2. Повторювати кроки 3 та 4, поки центроїди не зміниться.

Розширення класичного алгоритму кластеризації k-means, шляхом використання графів, дозволить зберігати інформацію, яка часто відкидається у більш простих моделях. Наприклад, представляючи веб-документи графами замість векторів, ми можемо зберігати таку інформацію, як порядок появи термінів або там, де в документі з’являються терміни. Це, в свою чергу, може призвести до поліпшення якості кластеризації.

Суть алгоритмів основаних на теорії графів полягає в тому, що вибірка об'єктів представляється у вигляді графа $G = (V, E)$, вершинам якого

відповідають об'єкти, а ребра мають вагу, рівний «відстані» між об'єктами. Перевагою графових алгоритмів кластеризації є наочність, відносна простота реалізації і можливість внесення різних удосконалень, заснованих на геометричних міркуваннях.

Існує пряма залежність між відстанню редагування графів та максимальним загальним підграфом між двома графами. Зокрема, вони є рівнозначними за певних обмежень щодо вагових функцій. Припустимо, що граф g – це максимальний загальний підграф (mcs) графів G_1 і G_2, тобто: $g = mcs(G_1, G_2)$, якщо $g \subseteq G_1, g \subseteq G_2$ та не існує жодного іншого підграфу $g' = (g' \subseteq G_1, g' \subseteq G_2)$, такого що $|g'| > |g|$.

Аналогічно, існує додаткова ідея мінімального загального суперграфа. Граф g – мінімальний загальний суперграф (MCS) графів G_1 і G_2, тобто $g = MCS(G_1, G_2)$, якщо $G_1 \subseteq g, G_2 \subseteq g$ та не існує жодного іншого суперграфу $g' = (G_1 \subseteq g', G_2 \subseteq g')$, такого що $|g'| < |g|$.

Загальний підхід полягає в тому, щоб створити граф сумісності для двох заданих графів, а потім знайти найбільше спільне в ньому. При обчисленні функції відповідності редагування на основі відстані редагування графів функція з найнижчою вагою еквівалентна максимальному загальному підграфу між двома графами за певних умов на коефіцієнтах ваг. Отже, загальний підграф – це частина обох графів, яка не змінюється, видаляючи чи вставляючи вузли та ребра. Для модифікації графа G_1 у G_2 потрібно лише виконати наступні дії:

1. Видалити вузли та ребра з G_1, які не відображаються у $mcs(G_1, G_2)$.
2. Виконати будь-які підстановки вузлів або ребер.
3. Додати вузли та ребра з G_2, які не відображаються у мс $mcs(G_1, G_2)$.

Згідно з даним спостереженням, розмір максимального загального підграфа пов'язаний з подібністю двох графів, було запроваджено міру відстаней на основі mcs. Визначено таку міру відстані:

$$d_{MCS}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{\max(|G_1|, |G_2|)}, \quad (1)$$

де $max\ (x, y)$ – звичайний максимум двох чисел x і y, і $|\ldots|$ вказує розмір графа (зазвичай прийнято вважати кількість вузлів у графі). Ця міра відстані має чотири важливі властивості:

- обмеження у формуванні числа в інтервалі [0,1];
- відстань дорівнює 0 лише тоді, коли два графи однакові;
- відстань між двома графами симетрична;
- відстань підкоряється нерівності трикутника, що забезпечує вимірювання відстані в інтуїтивно зрозумілому вигляді.

Перевага такого підходу перед методом відстані редагування графів полягає в тому, що він не потребує визначення жодних коефіцієнтів витрат або інших параметрів.

Поняття медіани набору графів, діє як представник множини. Медіана набору графів S є графом $g \in S$ таким, що g має найменшу середню відстань до всіх елементів у S:

$$\mathrm{g} = \arg\min_{\forall s \in S} \left(\frac{1}{|S|} \sum_{i=1}^{|S|} d(s, G_i) \right). \qquad (2)$$

Оскільки $g \in S$, то просто обчислити середню відстань до всіх графів для кожного графа в S. Медіана набору графів існує завжди, при чому може бути середнім значенням, а може і не бути.

Тепер, коли у є міра відстані для графів (рівняння 1) та метод визначення представника набору графів (медіана, рівняння 2), можна застосувати той самий метод до наборів даних, елементами яких є графи, а не вектори. Таким чином, необхідно:

- замінити вимірювання відстані, використовуваного на кроці 3, графовим вимірюванням відстані;
- заміною центроїд, обчислений на етапі 2, медіаною набору графів.

Модифікований алгоритм *k*-means з використанням теорії графів наведено у табл. 2.

Таблиця 2

Модифікований алгоритм кластеризації *k*-means

Вхідні дані:	сукупність *n* елементів даних (представлених графами) та параметр *k*, що визначає кількість кластерів
Вихідні дані:	центроїди кластерів (представлені серединними графами) і кластер (ціле число в [1, *k*]), для кожного елемента даних воно належить
Крок 1.	Призначити кожен елемент даних випадковим чином кластеру (від 1 до *k*).
Крок 2.	Використовуючи початкове призначення, визначити медіану набору графіків кожного кластеру.
Крок 3.	З огляду на нові медіани, призначити кожен елемент даних в кластері його найближчої медіани, використовуючи графову міру відстані.
Крок 4.	Повторно обчислити медіани, як на кроці 2. Повторити кроки 3 та 4, поки медіани не зміняться

2.3. Оптимізація початкового числа кластерів

Для проблеми автоматичного визначення оптимальної кількості кластерів необхідно знати індекс валідації кластера, який є показником якості кластеризації. Індекс Данна є одним з таких показників, однак він чутливий до шуму та забруднень. Спочатку маємо індекс *C*, який визначається як:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}, \quad (3)$$

де *S* – сума всіх відстаней пар елементів в одному кластері. Визначимо, що *l* – кількість пар, що використовуються для обчислення *S*. S_{min} і S_{max} – це сума *l* найменшої та найбільшої відстаней відповідно. Чим менше значення *C*, тим краще кластеризація.

Таблиця 3

Результати для *F*-серій при використанні *k*-means

Кількість кластерів	Індекс Данна	Індекс Девіса-Боулдіна	Індекс Гудмана-Краскала	С індекс	Рандомний індекс
2	0.6667	1.8567	0.3431	0.2451	0.5304
3	0.6667	1.8665	0.5163	0.3687	0.6604
4	0.6667	1.7833	0.6161	0.4188	0.7281
5	0.6667	1.7785	0.6795	0.4391	0.7578
6	0.6667	1.7091	0.7207	0.4028	0.7665
7	0.6667	1.6713	0.7588	0.3745	0.7775
8	0.6667	1.7688	0.7557	0.3780	0.7695
9	0.6667	1.6971	0.7956	0.3385	0.7761
10	0.6667	1.6560	0.8109	0.3236	0.7779

Інший показник дійсності – індекс Девіс-Боулдіна, визначений як:

$$DB = \frac{1}{M}\sum_{i=1}^{M} \max_{j=1,\dots,M;\ j\neq i}(d_{ij}), \tag{4}$$

де M – кількість кластерів та:

$$d_{ij} = \frac{r_i + r_j}{d(c_i, c_j)}. \tag{5}$$

В даному рівнянні r_i – середня відстань усіх елементів даних кластера i до їх центру, $d(c_i, c_j)$ – відстань між центрами кластерів i та j. Міра d_{ij}, подібна до індексу Данна, за компактністю (чисельник) та розділенням (знаменник) кластерних пар. Бажано невелике значення індексу Девіса-Боулдіна.

Таблиця 4

Результати для *F*-серій використовуючи випадкову ініціалізацію

Кількість кластерів	Індекс Данна	Індекс Девіса-Боулдіна	Індекс Гудмана-Краскала	С індекс	Рандомний індекс

2	0.6667	1.8567	0.3431	0.2451	0.5304
3	0.6667	1.8660	0.5440	0.3480	0.6676
4	0.6667	1.7983	0.6202	0.4173	0.7169
5	0.6667	1.9934	0.5629	0.4889	0.7057
6	0.6818	1.7981	0.6818	0.4244	0.7644
7	0.6667	1.8774	0.6772	0.4545	0.7634
8	0.6471	1.9110	0.6615	0.4763	0.7695
9	0.6667	1.6304	0.7472	0.4011	0.7831
10	0.6667	1.7314	0.7086	0.4610	0.7751

Також, існує індекс Гудмена-Крускала, який за принципом ініціалізації даних схожий на ініціалізацію при випадковому індексі, проте має ряд відмінностей. У методі Гудмена-Крускала розглядаються всі чотири групи даних (q, r, s, t) і перевіряються, чи відповідають вони одному з наступних випадків:

1. $d(q,r) < d(s,t)$; q і r в одному кластері; s і t в різних кластерах.
2. $d(q,r) > d(s,t)$; q і r в різних кластерах; s і t в одному кластері.
3. $d(q,r) < d(s,t)$; q і r в різних кластерах; s і t в одному кластері.
4. $d(q,r) > d(s,t)$; q і r в одному кластері; s і t в різних кластерах.

Якщо виконується перший або другий випадок то, це вказує на те, що пари елементів, що знаходяться в одному кластері, повинні мати меншу відстань, ніж пари елементів, які знаходяться в різних кластерах. Аналогічно для третього і четвертого випадків, які називають розбіжними.

Нехай S^+ – кількість суперечливих четвірок, а S^- – кількість невідповідних четвірок. Відповідно, індекс Гудмена-Крускала можна задати, як:

$$GK = \frac{S^+ - S^-}{S^+ + S^-}. \quad (6)$$

Велике значення індексу *GK* вказує на гарну кластеризацію. Проблема даного методу заключається в тому, що складність обчислення *GK* становить $\mathcal{O}(n^4)$, де n – кількість елементів у наборі даних. Таким чином, обчислення цього індексу може зайняти більше часу, ніж виконання кластеризації.

Було проведено експерименти зі значеннями *k*, що змінюються від 2 до 10 для *F*-серій і наборів даних серії *J* при використанні як глобальних засобів *k*-means, так і випадкової ініціалізації.

Випадкова ініціалізація здійснюється шляхом випадкового присвоєння кожного елемента даних кластеру. Неможливо повторно використовувати одну і ту ж випадкову ініціалізацію для різних значень *k*, тому кожен експеримент має окрему випадкову ініціалізацію. Результати представлені в табл. 3-6. Найкраща кількість кластерів визначається з випадкового індексу та взаємної інформації, які вказують на ефективність порівняно з первинними значеннями.

З результатів видно, що ці два індекси однакові для *k* експериментів, що використовували глобальний *k*-means. Для експериментів, що використовують випадкову ініціалізацію, не було домовленостей, і, отже, не можна остаточно визначитися з найкращою кількістю кластерів у цих випадках.

Таблиця 5

Результати для *J*-серій при використанні *k*-means

Кількість кластерів	Індекс Данна	Індекс Девіса-Боулдіна	Індекс Гудмана-Краскала	C індекс	Рандомний індекс
2	0.4286	1.9427	0.2376	0.2950	0.4850
3	0.6250	1.8845	0.4544	0.3987	0.6830
4	0.6500	1.8174	0.5328	0.4338	0.7618
5	0.6000	1.7792	0.5797	0.4122	0.7986
6	0.6000	1.7768	0.6612	0.3610	0.8471

7	0.6000	1.7653	0.6692	0.3612	0.8638
8	0.6154	1.7453	0.7300	0.3177	0.8833
9	0.6154	1.7612	0.7543	0.2947	0.8978
10	0.6154	1.7379	0.7686	0.2855	0.9049

Наступним спостереженням щодо результатів є те, що індекси Девіса-Боулдіна, і Гудмана-Крускала завжди збігались на одному значенні *k*. Крім того, узгодження цих двох індексів ефективності збігається з оптимальними значеннями для випадкового індексу та взаємної інформації, визначеними за допомогою *k*-means.

Індекс Данна і *C* не здається дуже корисним з точки зору пошуку правильного значення *k*. Лише у випадку глобального *k*-means із серії *F*, індекс Данна збігався з іншими індексами, і навіть тоді індекс Данна мав однакові значення для всіх *k*. Індекс *C* пройшов трохи краще, співпавши з іншими індексами *k*-means для *J*-серій; хоча оптимальне значення було при $k = 2$ для *k*-means із серії *F*.

Таблиця 6

Результати для *J*-серій використовуючи випадкову ініціалізацію

Кількість кластерів	Індекс Данна	Індекс Девіса-Боулдіна	Індекс Гудмана-Краскала	С індекс	Рандомний індекс
2	0.4286	1.9427	0.2376	0.2950	0.4850
3	0.4286	1.9337	0.1970	0.4466	0.6604
4	0.5833	1.8693	0.3849	0.4830	0.7205
5	0.6500	1.7627	0.5727	0.4123	0.7907
6	0.5714	1.8642	0.5313	0.4680	0.7772
7	0.6000	1.9639	0.6046	0.4043	0.8360

8	0.5833	1.8532	0.6180	0.4138	0.8617
9	0.5833	1.9477	0.6163	0.4066	0.8581
10	0.5714	1.8726	0.6063	0.4800	0.8659

2.4. Особливості реалізації модифікованого алгоритму

Отже, з огляду на запропоновану модифікацію, можна зробити висновок, що реалізація модифікованого методу кластеризації пошукових запитів користувачів потребує алгоритмів та структур даних, що забезпечують швидкий пошук на великих об’ємах повідомлень та пакетів. Задача пошуку інформації в програмній інженерії є достатньо складною і актуальною задачею, тому варто зупинитись на описі цієї задачі.

Також, необхідно звернути увагу на те, що результат кластеризації має бути максимально зрозумілим для користувача, оскільки з запропонованих аналогів жоден не володіє такою характеристикою.

Застосування кешування може прискорити роботу алгоритму. Необхідна зміна структури даних, так як операція кластеризації може займати надзвичайно багато часу в рамках пошуку цільового вузла. Необхідно також використовувати новий підхід що до обробки нових структур згенерованих даних.

2.5. Висновки до розділу 2

В даному розділі було проведено ґрунтовну роботу в напрямку дослідження розробки методу кластеризації пошукових запитів користувачів та аналіз способів покращення початкової кількості кластерів. Було виконане наступне:

- вибір базового методу;
- дослідження можливої модифікації обраного алгоритму;
- розробка модифікації методу;
- покращення кількості кластерів;

– аналіз можливих особливостей реалізацій модифікованого методу.

Модифікація методу *k*-means полягає в тому, що у порівнянні з базовим методом необхідно замінити вимірювання евклідової відстані, графовим вимврюванням, а також замінити центроїд на медіану набору графів.

Про особливості реалізації модифікованого алгоритму кластеризації та розробку програмного застосунку для вирішення поставленої задачі описано у розділі 3 даної дисертації.

3. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ МОДИФІКОВАНОГО МЕТОДУ КЛАСТЕРИЗАЦІЇ

Даний розділ дисертації присвячений детальному аналізу:

- засобів розробки програмного додатку;
- архітектури розроблюваної системи;
- реалізованих методів.

3.1. Аналіз засобів розробки програмного забезпечення

З огляду на те, що поставлену задачу зручніше всього виконувати у браузері, було вирішено реалізувати програмне забезпечення у вигляді веб застосунку, що має усі компоненти, необхідні для повноцінної роботи застосунку.

Пропонується розглянути найпопулярніші інструменти для створення веб-додатків.

3.1.1. *Python*

Python – це інтерпретована, спочатку об'єктно-орієнтованою мова програмування. Вона надзвичайно проста і містить невелику кількість ключових слів, в той же час дуже гнучка і виразна. Це мова більш високого рівня ніж Pascal, C++ і C, що досягається, в основному, за рахунок вбудованих високорівневих структур даних (списки, словники).

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Безсумнівним достоїнством є те, що інтерпретатор Python реалізований практично на всіх платформах і операційних системах. Наявна розширюваність мови, тобто є можливість вдосконалення мови всіма усіма зацікавленими програмістами. Інтерпретатор написаний на C і вихідний код доступний для будь-яких маніпуляцій. Також, наявне велике число модулів, які підключаються до програми, що забезпечують різні додаткові можливості. Такі модулі пишуться на C і на самому Python. Як приклад можна навести такі модулі:

- Numerical Python – розширені математичні можливості, такі як маніпуляції з цілими векторами і матрицями;
- Tkinter – побудова додатків з використанням графічного інтерфейсу користувача (GUI) на основі широко розповсюдженого на X-Windows Tk-інтерфейсу;
- OpenGL – використання великої бібліотеки графічного моделювання дво- і тривимірних об'єктів Open Graphics Library фірми Silicon Graphics Inc. Даний стандарт підтримується, в тому числі, в таких поширених операційних системах як Microsoft Windows 95 OSR 2, 98 і Windows NT 4.0.

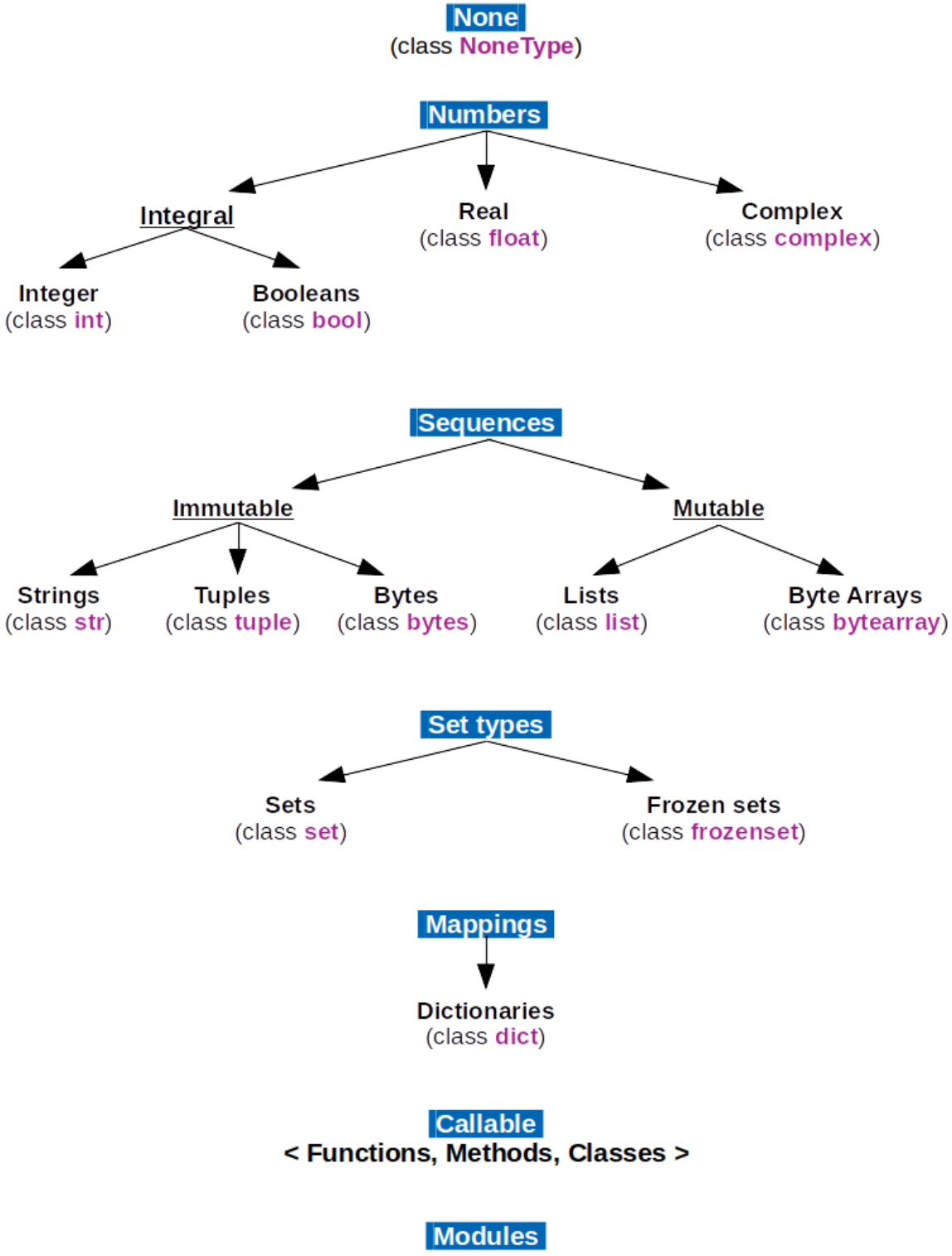


Рис. 6. Структура типів даних Python

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини. Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів. На рис. 6 зображено структуру типів даних мови програмування Python.

Багата стандартна бібліотека є однією з привабливих сторін Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML, тощо. Набір модулів для роботи з операційною системою дозволяє писати кросплатформні застосунки. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, юніт-тестуванням, серіалізацією даних та ін.

3.1.2. *Scikit-learn API*

Scikit-learn – це вільна бібліотека машинного навчання для мови програмування Python. В даному API представлені різні алгоритми класифікації, регресії та кластеризації, включаючи підтримуючі векторні машини, випадкові ліси, збільшення градієнта, *k*-means та DBSCAN, та є розробленим для взаємодії з числовими та науковими бібліотеками Python: NumPy та SciPy.

Проект scikit-learn розпочався як scikits.learn, проект є окремо розробленим та розповсюдженим та стороннім розширенням для SciPy. Оригінальну кодову базу згодом переписали інші розробники. Scikit-learn – одна з найпопулярніших бібліотек машинного навчання на GitHub.

Scikit-learn в значній мірі написаний на Python і широко використовує numpy для високоефективних операцій лінійної алгебри та операцій з масивами. Крім того, деякі основні алгоритми написані в Cython для підвищення продуктивності. Векторні машини підтримки реалізовані

обгорткою Cython навколо LIBSVM; логістична регресія та лінійні підтримуючі векторні машини аналогічною обгорткою навколо LIBLINEAR. У таких випадках розширення цих методів за допомогою Python може бути неможливим.

Scikit-learn добре поєднується з багатьма іншими бібліотеками Python, такими як matplotlib і plotly для побудови графіків, numpy для матриці векторизації, панд даних фреймів, scipy та багато іншого.

3.1.3. *Django Chart.js*

Візуалізація результатів аналізу була виконана за допомогою бібліотеки Chart.js. Chart.js – бібліотека з відкритим кодом (вона доступна на GitHub), яка допомагає легко візуалізувати дані за допомогою JavaScript.
Це схоже на Chartist та Google Charts. Вона підтримує 8 різних типів графіків, і всі вони адаптивні.

Все, що потрібно для того, щоб намалювати графік використовуючи Chart.js – це:

- вказати де саме на сторінці слід відобразити графік;
- вказати який саме тип графіку необхідно намалювати;
- вказати вибірку даних.

Django Chartjs дозволяє керувати діаграмами у програмі створеній за допомогою Django API, що розроблена сумісно з бібліотеками Chart.js та Highcharts JS. Використовуючи набір заздалегідь визначених класів переглядів, можна розпочинати роботу одразу після написання SQL-запиту. Підключення та налаштування бібліотеки до розроблюваного проекту зображено на рис. 7 та 8. Приклад застосування бібліотеки наведено у лістингу 1.

```
pip install django-chartjs
```

Рис. 7. Підключення Django Chartjs до проекту

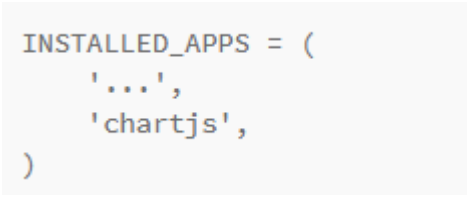


Рис. 8. Налаштування бібліотеки Django Chartjs

Лістинг 1. Встановлення Django Chartjs

```
{% load staticfiles %}
<html>
    <head>
        <title>django-chartjs line chart demo</title>
        <!--[if lte IE 8]>
            <script src="{% static 'js/excanvas.js' %}"></script>
        <![endif]-->
    </head>
    <body>
        <h1>Some Line Charts loaded in Ajax!</h1>

        <canvas id="myChart" width="500" height="400"></canvas>

        <script   type="text/javascript"   src="http://code.jquery.com/jquery-
1.10.0.min.js"></script>
        <script   type="text/javascript"   src="{%   static   'js/Chart.min.js'
%}"></script>
        <script type="text/javascript">
            $.get('{% url "line_chart_json" %}', function(data) {
                var ctx = $("#myChart").get(0).getContext("2d");
                new Chart(ctx, {
                    type: 'line', data: data
                });
            });
        </script>
    </body>
</html>
```

Таким чином, на сторінку візуалізації результатів аналізу був доданий HTML-елемент canvas. Цей елемент був доданий в специфікації HTML5 та використовується для того, щоб малювати графіки за допомогою JavaScript сценаріїв. Відповідні значення висоти та ширини були встановлені для елементу, а також елементу був присвоєний унікальний ідентифікатор (HTML-атрибут «id»), для того щоб ми могли вказати Chart.js який саме HTML-елемент використовувати для відображення графіків.

Так як нам необхідно візуалізувати скільки пошукових запитів було віднесено до кожного кластеру, було обрано тип діаграми, що має назву «Pie»

(рис. 9): діаграма поділяється на сегменти, дуга кожного сегмента показує пропорцію значень певного класу даних до загальної кількості значень. Цей тип діаграми відмінно демонструє пропорційні відношення між даними.

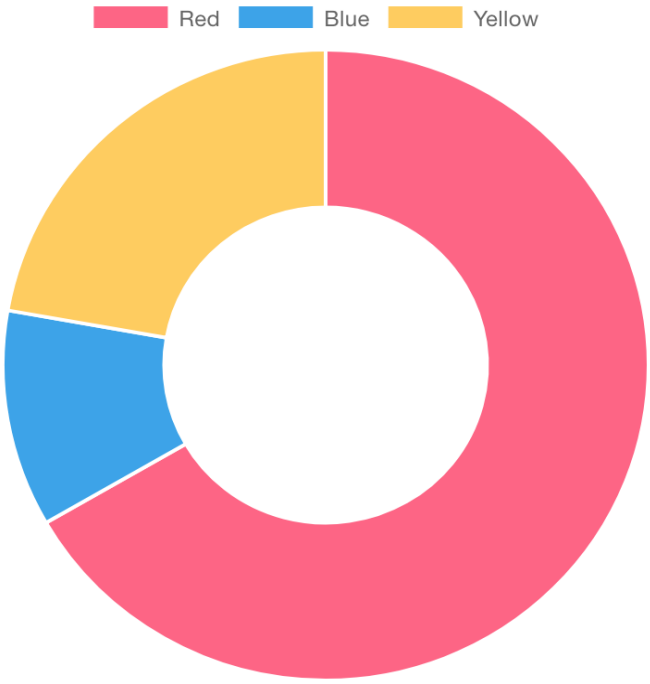


Рис. 9. Приклад діаграми «Pie»

3.2. Архітектура розробленого програмного забезпечення

З огляду на можливі області застосування даного алгоритму, було вирішено, що архітектура системи повинна бути реалізована на основі клієнт-серверної архітектури, а саме з використанням шаблону MVC.

MVC, Model-view-controller – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на

роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Елементи архітектури системи розглянуто далі в тексті та зображені на рис. 10:

1. Вигляд.
   - Bootstrap – набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків. Виконує роль клієнтського фреймворку, тобто інтерфейсу для користувача, на відміну від коду серверної сторони, який знаходиться на сервері.
   - Resources – ресурси додатку, що описують кнопки, назви табів, назви фонів, текстові константи, розширення, внутрішні зображення.

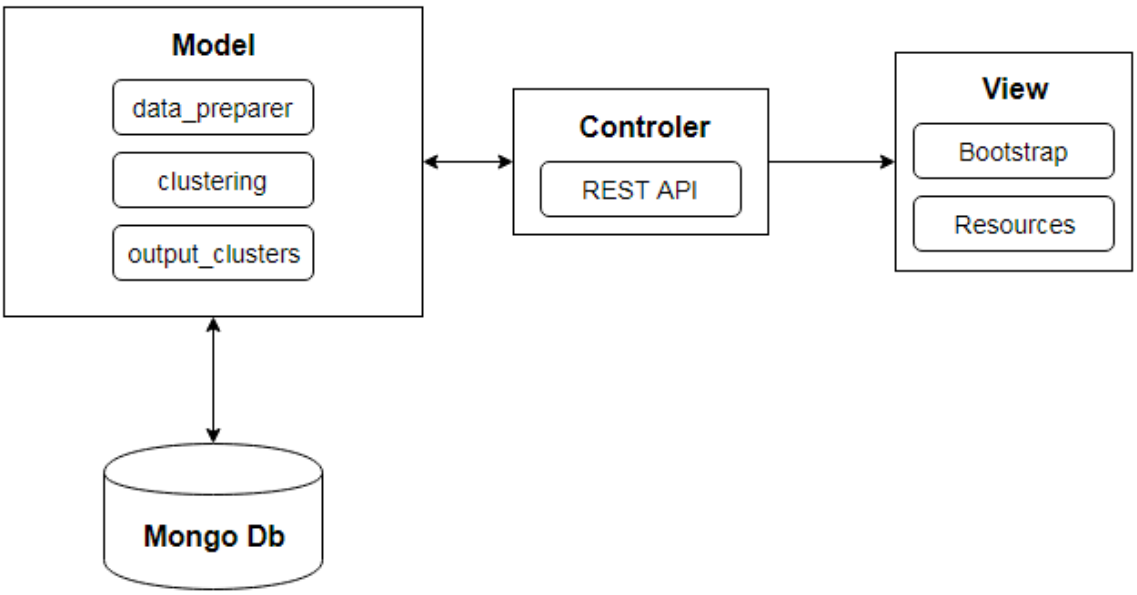


Рис. 10. Фізична архітектура мобільного додатку

2. Model – внутрішня логіка роботи застосунку, об’єднана з модулями розширеннями, базами даних, внутрішніх станів системи. Лістинги відповідних модулів знаходяться у додатку 1.
   - MongoDB – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів. MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій.

– data_preparer – модуль, що відповідає за приготування даних до кластеризації. А саме, відбувається порівняння введених користувачем ключових слів з дата сетами пошукових запитів (запити, що не мають збіжності з ключовими словами, відкидаються). На рис. 11 зображено структуру класу.

data_preparer	
+ __init__()	void
+ __read_dataset()	void
+ build_vocabulary()	void
+ generate_document_term_matrix()	void
+ apply_tf_idf()	void
+ __dict_to_sparse_matrix()	void

Рис. 11. Клас data_preparer

– output_clusters – модуль підготовки даних до візуалізації, тобто перетворює дані так, щоб їх можна було використовувати сумісно з бібліотекою Chart.js, що була описана підрозділі 3.1. Структуру класу зображено на рис. 12.

output_clusters	
+ __init__()	void
+ prepare_data()	void
+ write_to_csv_file()	function
+ store data()	void

Рис. 12. Клас output_clusters

– clustering – даний модуль виконує кластеризацію модифікованим методом, шляхом знаходження подібності між векторами речень. В тому випадку, коли коефіцієнт подібності перевищує порогове

значення, то дані речення позначаються як подібні і об’єднуються в кластер. На рис. 13 відображено структуру класу.

clustering_algorithm	
+ __init__()	void
+ find_sentence_similarity()	function
+ create_new_cluster()	function
+ cluster_sentences()	function
+ find_representative_sentence()	void

Рис. 13. Клас clustering_algorithm

3. Controler представлений у вигляді REST API, В основі REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP. Для даного додатку було розглянуто 3 типи повідомлень: GET, PUT, POST.

   Дана реалізація має такі переваги та покращення:

   – Рівень представлення даних являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і отримання від нього керуючих команд та даних.
   – Прикладний рівень реалізує основну логіку застосунку, на якому здійснюється необхідна обробка інформації;
   – Рівень доступу до даних забезпечує зберігання даних та доступ до них.

3.3. Особливості реалізації модифікованого алгоритму

Кластеризація даних була виконана за допомогою бібліотеки scikit learn, а саме з використанням модуля sklearn.cluster.

В даному модулі алгоритм *k*-means спрямований на вибір центроїдів, що мінімізують критерій інерції. Інерцію можна визнати мірою того, наскільки внутрішньо узгоджені кластери. Він страждає від різних недоліків:

– інерція робить припущення, що скупчення є опуклими і ізотропними, що не завжди так. Він погано реагує на подовжені скупчення або колектори неправильної форми;
– інерція не є нормованою метрикою: ми просто знаємо, що нижчі значення кращі, а нуль – оптимальний. Використання в модулі алгоритму зменшення розмірності, такого як аналіз основних компонентів до кластеризації *k*-means, полегшує цю проблему та пришвидшує обчислення.

Алгоритм підтримує вибіркові ваги, які можуть бути задані параметром sample_weight. Це дозволяє присвоїти більше ваги деяким зразкам при обчисленні центрів кластерів. Наприклад, присвоєння ваги 2 вибірці еквівалентно додаванню дубліката цього зразка до набору даних.

Може бути заданий параметр, який дозволить *k*-means працювати паралельно. Надаючи цьому параметру позитивне значення, використовується багато процесорів (за замовчуванням: 1). Значення -1 використовує всі доступні процесори, а -2 використовується на один менше. Паралелізація, як правило, прискорює обчислення за рахунок пам'яті (у цьому випадку потрібно зберігати кілька копій центроїдів, по одній для кожного завдання).

Варто зазначити, що на OS X при паралелізації *k*-means, коли numpy використовує Accelerate Framework буде отримано помилку. Така поведінка є очікуваною, оскільки Accelerate можна виконати, як підпроцес за допомогою бінарного файлу Python (що багатопроцесорна система не може зробити під даною позицією).

Лістинг 2. Застосування scikit-learn для реалізації алгоритму

```
from time import time
import numpy as np
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
```

```
def bench_k_means(estimator, name, data):
    t0 = time()
    estimator.fit(data)
    print('%-9s\t%.2fs\t%i\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f'
          % (name, (time() - t0), estimator.inertia_,
             metrics.homogeneity_score(labels, estimator.labels_),
             metrics.completeness_score(labels, estimator.labels_),
             metrics.v_measure_score(labels, estimator.labels_),
             metrics.adjusted_rand_score(labels, estimator.labels_),
             metrics.adjusted_mutual_info_score(labels,  estimator.labels_),
             metrics.silhouette_score(data, estimator.labels_,
                                      metric='euclidean',
                                      sample_size=sample_size)))

bench_k_means(KMeans(init='k-means++', n_clusters=n_digits, n_init=10),
              name="k-means++", data=data)
```

3.4. Аналіз розробленої системи

Розроблене програмне забезпечення дозволяє виконувати кластеризацію пошукових запитів відповідно до семантичного ядра сайту. Розроблений веб-застосунок має такі основні сторінки:

1. Реєстрація / авторизація. Починаючи свою роботу в веб-додатку користувач має авторизуватись для того, щоб можна було зберегти результат кластеризації та семантичне ядро сайту користувача, з метою розширення початкового дата сету.
2. Головна сторінка. Після успішної авторизації користувачеві відкривається головна сторінка (рис. 14), на якій розміщене вікно для введення ключових слів сайту користувача, або ж користувач має можливість обрати файл формату .txt або .xlsx.

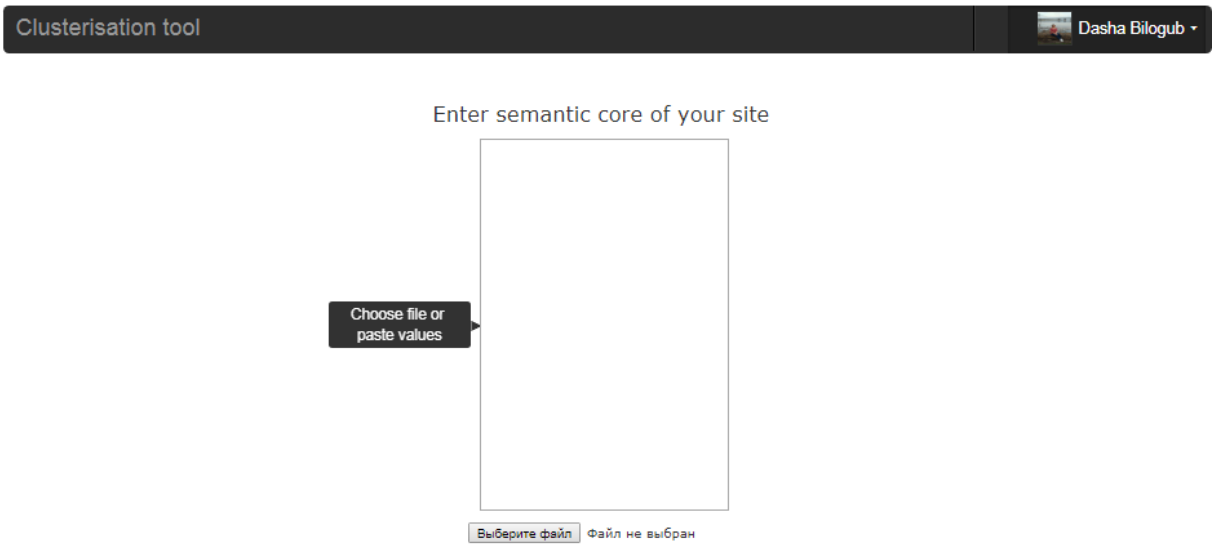


Рис. 14. Головна сторінка сайту

В якості подальшої модифікації веб-додатку планується розробка парсеру, який сам визначатиме ключові слова сайту, тобто користувачеві буде необхідно лише вставити посилання на свій сайт у відповідну форму.

3. Після того, як було введено семантичне ядро сайту, який необхідно проаналізувати, головна сторінка оновлюється. На сторінці відображається результат кластерзації, також, користувач має можливість завантажити результат аналізу у форматі .csv. Головну сторінку після аналізу зображено на рис. 15.

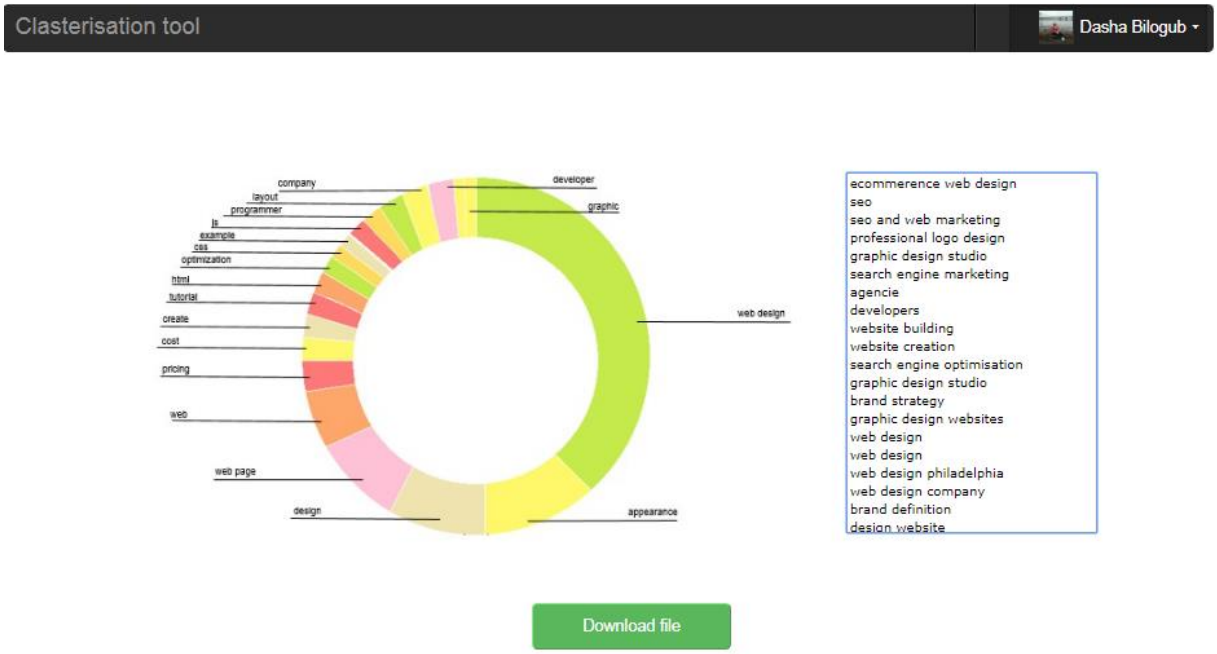


Рис. 15. Результат роботи алгоритму

3.5. Висновки до розділу 3

У рамках дослідження можливостей модифікації алгоритмів кластеризації пошукових запитів користувачів було розроблено покращення щодо точності кластеризації та початкового числа кластерів.

Для розробки застосунку було обрано мову програмування Python та нереляційну базу даних MongoDb, що зберігає дані у форматі JSON. Модифікований алгоритм кластеризації пошукових запитів було використано Scikit-learn API, що містить модулі, які полегшують реалізацію поставленої задачі. Дизайн розробленого веб-додатку було розроблено за допомогою фреймворку Bootstrap, а для зручної візуалізації результату аналізу було застосовано Chart.js.

Отже, модифікований алгоритм кластеризації пошукових запитів користувачів для веб-додатків було розроблено за рахунок створення та використання таких структурних компонент, як:

– Компонент відображення користувацького інтерфейсу з використанням Bootstrap та Chart.js.

– Компоненти логіки застосунку, які виконують підготовку даних до аналізу, кластеризацію та підготовку даних до візуалізації, а також відповідають за доступ системи до бази даних.
– Контролери, які відповідають за передачу даних між клієнтом та сервером.

Завдяки взаємодії цих компонентів в результаті було отримано такий веб-додаток, який може працювати у різноманітних браузерах, які надають операційні системи, без втрати сумісності з веб-середовищем.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Методика оцінювання ефективності алгоритмів кластеризації

Для оцінки якості роботи алгоритму кластеризації пошукових запитів користувачів необхідно дослідити на визначити ключові параметри, що є важливими для роботи самого алгоритму.

Після детального вивчення питання можна ввести наступні показники:

- кількість вершин графа;
- ступінь збіжності вузлів;
- швидкодія алгоритму.

Дані характеристики максимально повно оцінюють покращення в рамках модифікованого алгоритму кластеризації пошукових запитів користувачів, у порівнянні з базовим.

Намагаючись дотримуватися методології експериментів, було вибрано вибірку з 5000 пошукових запитів та зафіксовано бажану кількість кластерів, а саме $k = 40$. Змінення результатів між прогонами відбувається від випадкової ініціалізації на першому кроці алгоритму *k*-means. Для цих експериментів ми використовували вимірювання відстані та стандартне подання, яке описане в другому розділі.

4.2. Результат роботи модифікованого алгоритму

Взаємна інформація являє собою загальну ступінь узгодженості, що забезпечується істиною, з перевагою кластерів, які мають високу чистоту (тобто є однорідними щодо класів об'єднаних об’єктів). Більш високі значення означають кращі показники роботи алгоритму кластеризації.

З рис. 16 ми бачимо, що взаємна інформація, як правило, має тенденцію до збільшення, оскільки допускаються все більші та більші графи. Це має сенс, оскільки більші графи містять більше інформації.

Таблиця 7

Порівняння базового та модифікованого методів

Метод обчислення відстані	**Максимальна кількість вершин графа**	**Ступінь збіжності**
Графова міра відстані	150	0.2218
	120	0.2142
	90	0.2074
	75	0.2045
	60	0.1865
	45	0.1758
	30	0.1617
	15	0.1540
	5	0.1326
Евлідова відстань	–	0.046

Використання евклідової відстані в алгоритмі *k*-means дає гірші результати, оскільки вона не має властивості інваріантності довжини вектора. Через це документи зі схожими співвідношеннями частоти ключових слів, але відмінними в загальній частоті мають великі відстані між кластерами, навіть якщо вони вважаються подібними.

З графіку на рис. 16 видно, що навіть маючи лише 5 вузлів, модифікований метод перевершує як *k*-means з використанням евклідової відстані, так і випадкову базову лінію; при збільшенні кількості вузлів на графіку, продуктивність наближається до показників ефективності інших методів *k*-means, поки вона навіть не перевищила показники найкращого методу *k*-means, при 75 та більше вузлах.

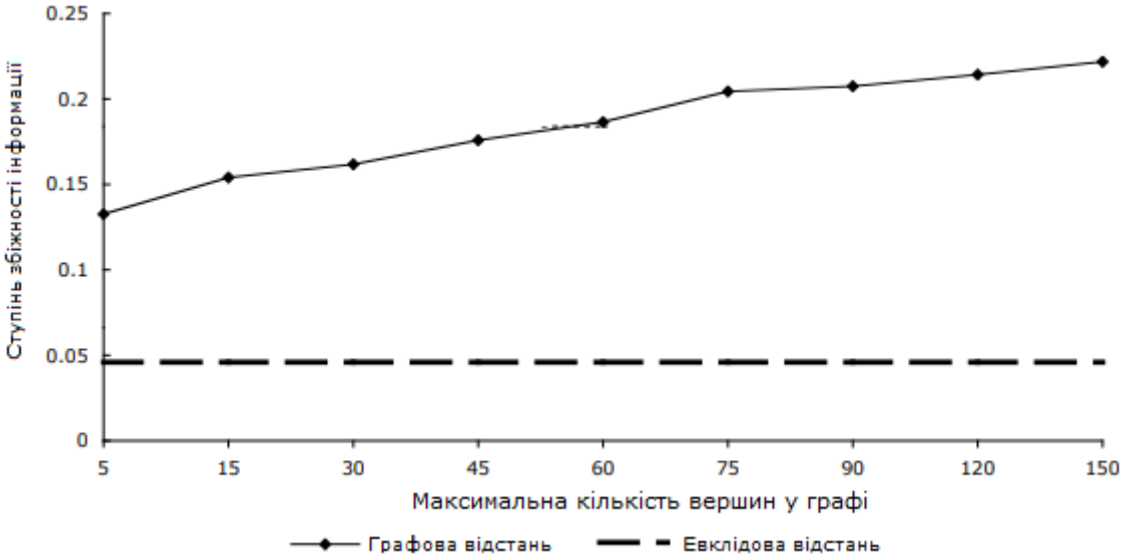


Рис. 16. Діаграма порівняння результатів використання традиційного та модифікованого алгоритму за подібністю даних у кластері

Однак підвищення продуктивності не завжди суворо пропорційно збільшенню розміру графіка. Наприклад, поліпшення від 60 до 75 більше, ніж покращення від 75 до 90, навіть при додаванні 15 нових вузлів у кожному випадку. Це може бути пов’язано з тим, що додаткові вузли, що додаються, коли ми збільшуємо розмір графа, хоча вони часто зустрічаються, не завжди можуть надавати інформацію, яка корисна для розмежування між фразами, а фактично можуть заважати виконанню введення сторонніх даних.

Подальше вдосконалення може полягати в пошуку кращих методів вибору вузлів, які будуть використовуватись у кожному графіку, а не строго покладатися на частоту терміна.

4.3. Висновки до розділу 4

У даному розділі було описано методологію тестування для розроблюваної системи, проаналізовано результати роботи системи у майбутньому.

За результати тестування можна заявити, що розроблена система не має помилок, які можуть призвести до виключних ситуацій при роботі користувача.

Також, за результатами аналізу було виявлено, що система повністю відповідає заявленим вимогам, що були висунуті під час аналізу систем-аналогів.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1. Виділення проблеми

Для пошуку інформації за допомогою пошукової системи користувач формулює запит. Робота пошукової системи полягає в тому, щоб за запитом користувача знайти документи, що містять або зазначені ключові слова, або слова, будь-яким чином пов'язані з ключовими словами . При цьому пошукова система генерує сторінку результатів пошуку. Така пошукова видача може містити різні типи результатів, наприклад: веб-сторінки, зображення, аудіофайли.

Ключове слово – слово в тексті, здатне в сукупності з іншими ключовими словами дати високорівневий опис змісту текстового документа, що дозволяє виявити його тематику. В інтернеті використовується головним чином для пошуку.

У HTML для завдання ключових слів є елементи HTML meta з атрибутом keywords. Такий шлях завдання ключових слів відкриває ще більше можливостей для зловживання, тому деякі пошукові системи використовують цей тег як фактор для поліпшення ранжирування сторінок, а деякі ні. Так наприклад, Google часто ігнорує ключові слова в тезі через занадто великого зловживання ним в минулому [31]. Однак їх використовують інші призначені для користувача агенти (наприклад, веб-браузери для пошуку по закладках).

Таким чином, при створення веб-сторінки виникає питання чи правильно зазначені ключові слова? Чи з’являється веб-сторінка при пошукових запитах, що безпосередньо відносяться до даної сторінки? Чи всі користувачі, які шукають однотипний об'єкт, потраплять на ту сторінку, де цей об'єкт представлений – тобто чи всі запити можуть просуватися на одній сторінці.

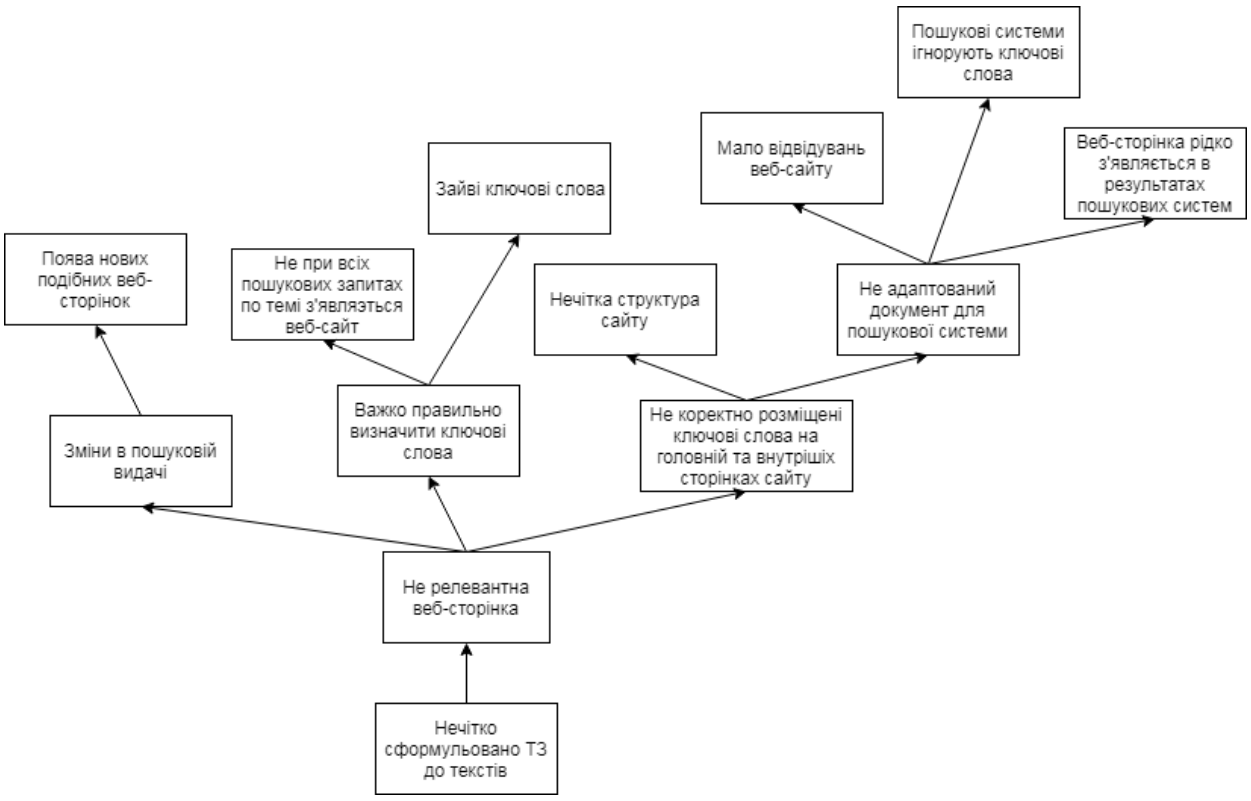


Рис. 17. Дерево проблем

Грамотне використання кластеризації допоможе підвищити релевантність документів. Кластеризація потрібна не тільки для адаптації документа відповідно до алгоритму роботи пошукової системи. Вона також допомагає самому SEO-фахівцю оцінити документ з технічної точки зору і об'єктивно поглянути на його недоліки. Таким чином, кластеризація підвищує якість роботи SEO-фахівця і проекту в цілому.

Кластеризація дає чітке уявлення про кількість і вигляд ключових фраз, які використовуються для просування сайту, що дозволяє більш грамотно формулювати технічні завдання до текстів.

А ще кластеризація допомагає побачити структуру сайту. Визначивши фрази для головних і внутрішніх сторінок, SEO-фахівець отримує точне уявлення про те, де і яка інформація знаходиться, як саме вона постає перед потенційними відвідувачами [32].

Основна проблема пов'язана з тим, що неможливо розробити семантичне ядро один раз і більше не повертатися до цього питання. І ось чому:

– нелогічність. Подання про угруповання запитів на основі пошукової видачі і логіки не завжди збігаються. Практично завжди SEO-фахівцю необхідно коригувати отримане угруповання: об'єднувати різні групи запитів в одну або ж розбивати одну на кілька. При цьому, існує можливість додавання зайвих запитів в групу або ж навпаки, упустити щось;
– чутливість до зміни пошукової видачі, що відбувається досить часто. Якщо згрупувати одне і те ж семантичне ядро в різний час, то результат може відрізнятися. Це пов'язано з тим, що органічна видача найчастіше нестабільна і відбувається перетасування сайтів.

Вище описані проблеми можна більш детально розглянути у схемі «Дерево проблем», яка зображена на рис. 17.

5.2. Зацікавлені сторони

У вирішенні вищеописаної проблеми існує кілька зацікавлених сторін.

В першу чергу, необхідно відмітити SEO-спеціалістів. Для того, щоб ресурс добре ранжирувався, SEO-спеціаліст упорядковує внутрішню структуру, файли, наповнює сторінки ключовими словами, а також працює над цитованістю бренду і його згадками. Результатом діяльності SEO-спеціаліста стає просування позицій різних сторінок сайту до перших місць видачі по певних запитах. Функції, що їх виконує SEO-спеціаліст, є похідними від факторів, які впливають на позиції сайту [33]:

1. Ключові слова. Пошуковики видають списки ресурсів у відповідь на запит користувача, сформульований словами. Щоб один сайт виявився вищим іншого, потрібно зібрати базу даних саме тих запитів (з їх частотністю і конкурентність), які цікавлять споживачів в першу чергу. А потім згрупувати і розташувати ці ключові слова по сторінках сайту так, щоб вони утворювали єдину смислову структуру. Незважаючи на те що ця частина роботи автоматизована, вона практично завжди вимагає «ручний» доопрацювання SEO-

спеціаліста. Ключові слова на сторінці стоять не самі по собі, а найчастіше є елементами тексту. Від того де, як і в якій кількості вони будуть розташовуватися, залежить реакція пошуковиків на контент.

2. Внутрішня оптимізація.
3. Зовнішня оптимізація.

Далі, варто відмітити, власників сайту або товару, що замовляють розробку веб-сторінки. Їм важливо, щоб їх сторінка з'являлась при різноманітних запитах користувачів, що відносяться до даної тематики. Звісно, саме для цього наймаються SEO-спеціалісти, проте, саме замовники вирішують на скільки їм необхідні послуги даного працівника. Замовники готові до того, щоб вкласти більше коштів при розробці проекту, для того щоб в кінці мати гарний готовий продукт.

Третю категорію, яку можна виділити – є самі пошукові системи. Їх розробники зацікавлені в тому, щоб алгоритм роботи пошукової системи надавав найбільш наближений результат для пошукового запити користувача. Пошукова система аналізує вміст кожної сторінки для подальшого індексування. Слова можуть бути вилучені із заголовків, тексту сторінки, її ключових слів або спеціальних полів – метатегів. Корисність пошукової системи залежить від релевантності знайдених нею сторінок. Хоч мільйони веб-сторінок і можуть включати якесь слово або фразу, але одні з них можуть бути більш релевантні, популярні або авторитетні, ніж інші. Більшість пошукових систем використовує методи ранжирування, щоб вивести в початок списку «кращі» результати. Пошукові системи вирішують, які сторінки більш релевантні, і в якому порядку повинні бути показані результати, по-різному.

Також, до зацікавлених сторін можна віднести звичайних користувачів пошукових систем. Вони хочуть, щоб пошукова система правильно розуміла їх запити та видавала лише найкращі результати. Проте, користувачі не мають ніякого впливу на релевантність веб-сторінок і т.д., вони лише можуть

слугувати фактором для сумлінної роботи SEO-спеціаліста, а також розробників пошукових систем.

У табл. 8 підсумовано все вищесказане, визначено інтереси зацікавлених сторін та їх вплив (міра зацікавленості у вирішенні наявних проблем).

Таблиця 8

Зацікавлені сторони

Зацікавлена сторона	**Інтерес зацікавленої особи**	**Вплив зацікавленої особи**	**Стратегії приваблення зацікавлених сторін**
SEO-спеціалісти	Покращення релевантності веб-сторінки. Допомога при виборі та розміщенні ключових слів на сторінці	Високий	Проведення презентацій для представників зацікавлених сторін. Створення привабливої цінової політики, наявність знижок на послуги
Пошукові системи	Адаптації документа відповідно до алгоритму роботи пошукової системи	Високий	
Замовники веб-сайтів	Багато відвідувань сайту	Середній	
Користувачі	Можливість отримати найкращі результати за пошуковим запитом	Низький	

5.3. Комерційне рішення. Основні характеристики

Відповідно до вище зазначених проблем, можна описати кінцевий продукт, що має їх вирішувати. Даний програмний продукт буде реалізовувати описаний у попередній розділах автоматизований метод кластеризації веб-сайтів на основі пошукових запитів користувачів. Даний метод дозволяє визначити при яких пошукових запитах буде відображатись веб-сторінка. На основі аналізу пошукових запитів користувачів, буде визначено найбільш підходящі ключові слова, а також надаватимуться рекомендації щодо їх

розміщення на головних та внутрішніх сторінка веб-сайту, згідно з роботою алгоритмів найпопулярніших пошукових систем.

Таким чином, буде підвищено релевантність веб-документів та покращено роботу пошукових систем. Клопітку працю SEO-спеціалістів можна буде замінити автоматизованою роботою даного додатку. Очевидно, що клієнтом даного продукту є пошукові системи, а співпраця буде побудована на моделі співробітництва бізнесу для бізнесу, або як він ще називається B2B [34].

Даний програмний продукт повинен легко інтегруватися з існуючими механізмами роботи пошукової системи (наприклад, плагін для веб браузер). При запуску веб-сайту, дане програмне забезпечення виконуватиме перевірку його релевантності та надаватиме зміст SEO-спеціалісту про те, як необхідно модернізувати веб-документ, а саме статистику по ключовим словам та їх розміщення на головних та внутрішніх сторінках. По суті, буде отримано такий самий результат як буде отримано, як би це робила людина, проте з економією часу.

Зважаючи на вище сказане, можна зробити висновок, що проблем з інтеграцією з існуючими системами виникати не повинно, або вони повинні швидко і легко вирішитися за допомогою невеликої кількості спеціалістів.

5.4. Конкурентні переваги рішення

З розвитком мережі Інтернет, розвивалась і дана сфера. Тому, на сьогоднішній день існує чимало подібних програмних рішень.

Було детально розглянуто та проаналізовано недоліки існуючих аналогів:

- у частини додатків деякі функції безкоштовні, а решта – платні. Інша частина додатків надає тимчасову безкоштовну пробну версію. Проте, як зазначалось раніше необхідно постійно перевіряти релевантність веб-сайту, оскільки результати видачі пошукової системи постійно змінюються;

- неточний результат. Для коректної роботи більшості аналогів необхідне втручання людини. Отже, даний процес не є повністю автоматизованим;
- незрозумілий користувацький інтерфейс. У більшості аналогів користувачі зустрічались з проблемою, що необхідно завантажувати певні додаткові дані, не зрозуміло як потім зберігається результат;
- низька швидкість роботи при обробці великої кількості даних. Безкоштовні аналоги вирішують вищезазначену задачу, в середньому за 4-5 годин. Платні: можуть виконати ту ж саму задачу за декілька хвилин. Велику роль в даній проблемі відіграє саме алгоритм кластеризації.

Отже, на основі даних недоліків було визначено основні конкурентні переваги програмного продукту, що розроблюється, а саме:

- зменшення вартості визначення ключових слів для веб сторінки в порівнянні з існуючими аналогами;
- наявність системи знижок при довготривалому використанні даного веб застосунку;
- збільшення швидкості аналізу, не залежно від кількості даних;
- використання такого методу кластеризації пошукових запитів, що не використовувався для вирішення даної задачі;
- розробка зручного та зрозумілого користувацького інтерфейсу, що частково досягається використанням нового методу кластеризації.

5.5. Клієнти. Сегменти ринку споживання

Як вже зазначалося вище, клієнтами даного програмного забезпечення є пошукові системи.

Пошукова система тим краще, чим більше документів, релевантних запиту користувача, вона буде повертати. Результати пошуку можуть ставати менш релевантними через особливості алгоритмів або внаслідок людського

фактора. Станом на 2015 рік найпопулярнішою пошуковою системою в світі є Google, однак є країни, де користувачі віддали перевагу іншим пошуковикам.

В кожному регіоні і навіть країні, є відомі всім сервіси для інтернет пошуку потрібної інформації, а також свої особисті сайти, які використовують тільки в межах цих країн. Загальний графік часткою Пошукових систем світу зображено на рис. 18 [35].

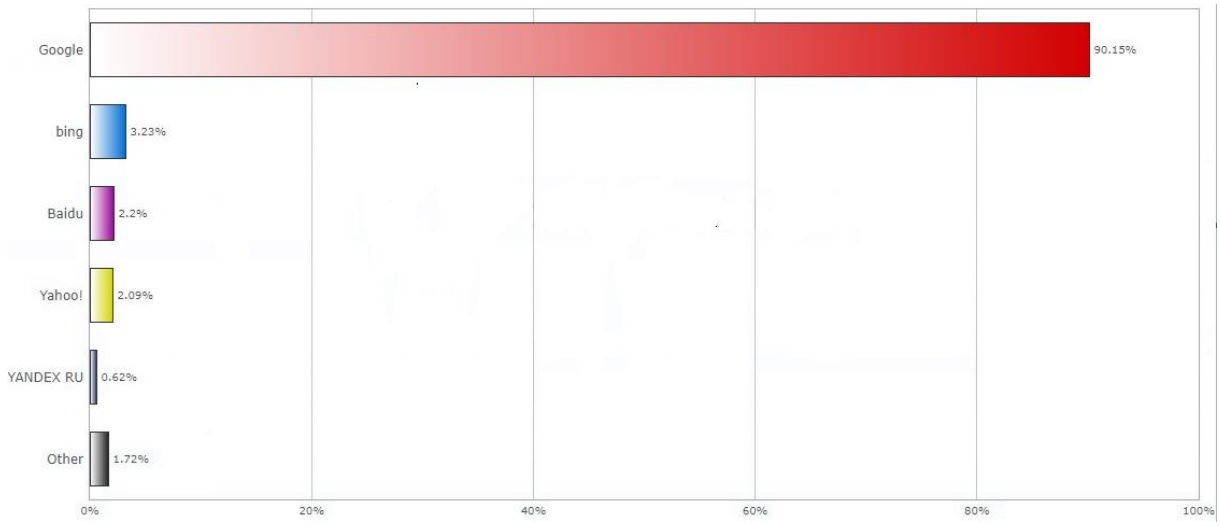


Рис. 18. Частка пошукових систем світу

Google, Bing, Yahoo! і ще деякі пошуковики використовуються по всьому світу, оскільки вони є міжнародними, тобто вони підтримують багато мов. Існують країни в яких на першому місці своя пошукова система, наприклад Росія (Яндекс, Mail.ru) і Китай (Soso). Незважаючи на те, що в Україні існують власні пошукові системи, такі як: UKR.NET, i.ua, meta-ukraine.com/ua/ та інші, більшість користувачів інтернету України віддають перевагу Google.

Таблиця 9

Статистика часток ринку пошукових систем

Пошукові системи України	**Частка користувачів, %**
Google	89.4
Yandex	7.09

Bing	1.21
Yahoo	0.84
Mail.ru	0.77
DuckDuckGo	0.41
Baidu	0.23
Other	0.06

Існує чотири типи пошукових систем:

- системи, що використовують пошукові роботи. Складаються з трьох частин: краулер («бот», «робот» або «павук»), індекс і програмне забезпечення пошукової системи. Краулер потрібен для обходу мережі і створення списків веб-сторінок. Індекс – великий архів копій веб-сторінок. Мета програмного забезпечення – оцінювати результати пошуку. Завдяки тому, що пошуковий робот в цьому механізмі постійно досліджує мережу, інформація більшою мірою актуальна. Більшість сучасних пошукових систем є системами даного типу;
- системи, керовані людиною (каталоги ресурсів). Ці пошукові системи одержують списки веб-сторінок. Каталог містить адресу, заголовок і короткий опис сайту. Каталог ресурсів шукає результати тільки з описів сторінки, представлених йому веб-майстрами. Гідність каталогів в тому, що всі ресурси перевіряються вручну, отже, і якість контенту буде краще. Але є і недолік – оновлення даних каталогів виконується вручну і може істотно відставати від реального стану справ. Ранжування сторінок не може миттєво змінюватися. Як приклади таких систем можна привести каталог Yahoo!, dmoz і Galaxy;

- гібридні системи. Такі пошукові системи, як Yahoo, Google, MSN, поєднують в собі функції систем, що використовують пошукові роботи, і систем, керованих людиною;
- мета-системи. Метапошукові системи об'єднують і ранжирують результати відразу декількох пошукових систем. Ці пошукові системи були корисні, коли у кожної пошукової системи був унікальний індекс, і пошукові системи були менш «розумними». Оскільки зараз пошук набагато покращився, потреба в них зменшилася. Приклади: MetaCrawler і MSN Search [36].

Статистика пошукових систем – предмет професійного інтересу самих різних груп користувачів, але, перш за все, вона може бути корисна для рекламодавців, творців інтернет-ресурсів і лінгвістів. Також, дана інформація актуальна для мого дослідження [37].

Найбільша в світі пошукова система Google надає відкритий доступ до своєї статистики запитів. На відміну від пошукових систем Яндекс та Rambler, кількісна статистика доступна в форматі csv, візуально статистика представляється лише відносно – у вигляді графіка. Звіти виділяються особливою подробицею: наприклад, крім звичайної статистики запитів користувачів, можна подивитися ступінь конкуренції рекламодавців за конкретний пошуковий запит, переглянути історію трафіку для обраних ключових слів; надається підказка можливо корисних мінус-слів. Також особливістю цього сервісу є те, що можна ознайомитися з реальною вартістю тих чи інших запитів, на яку повинен буде розраховувати рекламодавець для участі в партнерській програмі медіагіганта.

В особливий вид статистику відображають графіки Google Trends. Сервіс дозволяє вводити до 5 різних запитів, вивчати і порівнювати зміну інтересу до них в світі у вигляді графіка за минулі 2-3 роки.

Пошуковик Yahoo теж надає свою статистику. Дані звіти не відрізняються подробицею: єдине, що показує сервіс Overture – це кількість запитів за окремими словами, а також по словосполученням. Але, незважаючи

на меншу результативність, він має наступну перевагу в порівнянні з сервісом від Google: видає інформацію в цифровій формі, а не у вигляді графіка.

Також, серед інших великих пошукових сайтів – Wordtracker, який збирає інформацію про звернення до пошукових систем Dogpile.com і Metacrawler.com. Компанія Microsoft також надає подібний до Google Trends безкоштовний сервіс під маркою Keyword Forecast, де можна порівнювати відразу кілька ключових слів і зміну інтересу до них на графіку за рік.

Отже, для співпраці, найбільш перспективними є великі пошукові системи, що використовують пошукових роботів. Для поставленої задачі найкращим варіантом – є Google. Оскільки, згідно з поточною статистикою дана пошукова система є найуживанішою в Україні та надає підтримку на різних мовах. З точки зору розробки програмного застосунку, дана система надає статистику пошукових запитів, що є ключовим елементом в розробці веб-додатку.

5.6. Унікальна ціннісна пропозиція

Цінність – це розуміння користі, яку покупець отримує від продукту. Користь і є цінність.

Ціннісна пропозиція – це сукупність переваг, які компанія готова надати споживачу, якщо сформована проблема, яку вирішує продукт [38]. При формуванні ціннісної пропозиції, треба описувати не сам товар, а проблему яку він вирішує. Дане поняття можна описати за допомогою формули: Ціннісна пропозицію = Проблема + Рішення / Продукт.

У дереві проблем було виділено низку проблем, а у зацікавлених сторонах – було визначено очікування відповідних сторін від продукту. Пошукові системи бажають отримати економний та надійний спосіб підвищити точність роботи пошукових алгоритмів, що досягається за допомогою адаптації документа. SEO-спеціалісти хочуть покращити релевантність веб-сайту та підвищити його відвідуваність, шляхом виявлення ключових слів та їх коректного розміщення на сторінках сайту. Майже те ж саме вимагають замовники цих сайтів – впізнаваність продукції (послуг,

тощо), що рекламується на веб-сторінках. Щодо користувачів, то їм достатньо щоб на їх пошукові запити, були отримані необхідні результати, тобто коректна робота пошукової системи. Дійсно, запропоноване рішення, дозволяє частково задовольнити всі з наведених вище вимог зацікавлених сторін та вирішити наведені проблеми.

Отже, основною унікальною ціннісною пропозицією є кластеризація пошукових запитів користувачів методом, який не використовувався в існуючих реалізаціях вирішення поставленої задачею. Основною перевагою розроблюваного програмного забезпечення є розробка зручного та зрозумілого користувацького інтерфейсу, оскільки більшість з наведених вище аналогів мають дану проблему.

5.7. Доходи та витрати

Сумарний дохід складається як сума доходу на продаж ліцензії на використання програмного забезпечення та надання послуг на його підтримку.

Продаж програмного забезпечення планується шляхом продажу ліцензії на програмне забезпечення, тобто укладання угоди, яка надає право використовувати програмне забезпечення. Вид ліцензії, який планується продаватися, це commercial (комерційна) ліцензія. Дана ліцензія передбачає використання закритого, власницького або пропрієтарного ПЗ. Пропрієтарне програмне забезпечення [39] – це програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права [40]. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж [41]. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається

закритим. Проте, код власницького продукту може бути й відкритим, але власник може обмежити права користувача умовами ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути й безплатне (тобто, некомерційне) програмне забезпечення.

Саме тому ліцензія буде розповсюджуватися на комерційне програмне забезпечення.

Платна технічна підтримка включає в себе реакцію на запит споживача та усунення недоліків роботи програми в строки погоджені договором використання програмного забезпечення.

До витрат відноситься:

- утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат);
- утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги);
- податкові витрати;
- оплата послуг юриста, бухгалтера, прибиральниці.

Детальніше ознайомитися з витратами на реалізацію проекту та прогнозованим прибутками можна з табл. 10.

Таблиця 10

Витрати на реалізацію проекту

Найменування витрат	**1-й місяць, т. $**	**2-й місяць, т. $**	**3-й місяць, т. $**	**4-й місяць, т. $**	**5-й місяць, т. $**	**6-й місяць, т. $**
Загальні витрати	2	2	2	2	2	2
Заробітна плата	4	10	10	15	15	15
Витрати	**6**	**12**	**12**	**17**	**17**	**17**
Заплановані прибутки						

Результат (без оподаткування)	**–6**	**–12**	**–12**	**–17**	**–17**	**–17**

Продовження табл. 10

Найменування витрат	**7-й місяць, т. $**	**8-й місяць, т. $**	**9-й місяць, т. $**	**10-й місяць, т. $**	**11-й місяць, т. $**	**12-й місяць, т. $**
Загальні витрати	2	2	2	2	2	2
Заробітна плата	20	20	25	25	25	25
Витрати	**22**	**22**	**27**	**27**	**27**	**27**
Заплановані прибутки	**60**	**60**	**60**	**60**	**60**	**60**
Результат (без оподаткування)	**38**	**38**	**33**	**33**	**33**	**33**

5.8. Бізнес-модель

Узагальнимо, все написане вище у лаконічну бізнес-модель у вигляді lean canvas.

Споживачі: бізнеси, що отримують дохід через користувацькі застосунки.

Проблема: висока вартість розробки системи з виконанням таких блоків: віддалений сервер, мобільний додаток, веб-додаток; утримання користувачів у мобільному додатку; лояльність клієнтів бізнесів.

Рішення: розроблення крос-платформних користувацьких додатків.

Унікальна ціннісна пропозиція: система, що дозволяє на основі однієї кодової бази створити платформо-орієнтовані додатки, що представлятимуть цільову бізнес-логіку.

Потоки доходів: доходи від продажу ліцензій; доходи від підтримки програмного забезпечення.

Структура витрат: утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат); утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги); податкові витрати; оплата послуг юриста, бухгалтера, прибиральниці; витрати на маркетинг.

Таблиця 11

Канва бізнес-моделі

<table>
<tr>
<td rowspan="2">Проблема
висока вартість розробки системи
утримання користувачів у мобільному додатку
лояльність клієнтів</td>
<td>Рішення
створення крос-платформних додатків
використання бібліотеки, що пришвидшує розрахунки</td>
<td rowspan="2">Унікальна ціннісна пропозиція
система, що дозволяє на основі однієї кодової бази створити платформо-орієнтовані додатки, що представлятимуть цільову бізнес-логіку</td>
<td>Прихована перевага
легка інтеграція з існуючими платформами
підтримка більше 5-ти платформ вже зараз</td>
<td rowspan="2">Споживач
бізнеси, що отримують дохід через користувацькі додатки</td>
</tr>
<tr>
<td>Ключові метрики
кількість проданих ліцензій</td>
<td>Канали
через платформи замовлення робіт з ПЗ;
конференції;
бізнес-тренінги;
інші бізнеси</td>
</tr>
<tr>
<td colspan="3">Структура витрат
утримання персоналу;
утримання робочих;
податкові витрати;</td>
<td colspan="2">Потоки доходів
доходи від продажу ліцензій;
доходи від підтримки програмного забезпечення</td>
</tr>
</table>

витрати на послуги юриста, бухгалтера витрати на маркетинг	

Також в канву бізнес-моделі включаються структурні блоки: прихована перевага (перевага, яку не можливо скопіювати або купити), ключові метрики (основні показники, що вимірюються) та канали (шляхи до користувачів).

Канали: через платформи замовлення робіт з ПЗ; конференції; бізнес-тренінги; інші бізнеси.

Ключові метрики: кількість проданих ліцензій.

Прихована перевага: легка інтеграція з існуючими платформами; підтримка більше 5-ти платформ вже зараз.

Бізнес-модуль наведено у зведеному вигляді у табл. 11.

Отже, зважаючи на дані у табл. 11, можна зробити висновок, що запропонована бібліотека, що реалізує описаний у дисертацій метод пришвидшення розрахунків на стороні мобільних додатків та не блокування користувацького інтерфейсу, має перспективи у своїй подальшій реалізації. Звичайно, проведений аналіз не враховує всіх ризиків та факторів, таких як специфіка оподаткування у країні ведення бізнесу, проте навіть наявних досліджень достатньо, щоб прогнозувати комерційний успіх продукту та його окупаємось.

5.9. Висновки до розділу 5

У даному розділі було проведено аналіз поточної ситуації у кластеризації пошукових запитів, виявлено наявні проблеми та підсумовано їх у відповідному дереві проблем. Наряду з проблемами було виділено основні зацікавлені сторони у вирішенні існуючих недоліків, ступінь впливу даних сторін на вирішення проблем. Як наслідок було запропоновано комерційне рішення з конкурентними перевагами, що задовольняє інтереси зацікавлених осіб, та виділено унікальну ціннісну пропозицію запропонованого продукту.

Було проведено аналіз майбутніх клієнтів, досліджено сегменти ринку споживання. Це дозволило спрогнозувати потенційні доходи та витрати на реалізацію продукту. У результаті була описана бізнес-модель, що обґрунтовує доцільність реалізації даного продукту та прогнозує його потенційну окупаємість та прибутковість в подальшому.

ВИСНОВКИ

У даній магістерській дисертації було поставлено за мету дослідити існуючі методи кластеризації пошукових запитів користувачів. Результатом досліджень має бути розробка модифікованого методу кластеризації, який має вирішувати проблеми існуючих алгоритмів.

У першому розділі було проаналізовано існуючі алгоритми кластеризації та досліджено аналоги, що виконують поставлену задачу. Виявлено їх основні переваги та недоліки. На основі проведеного аналізу було сформовано вимоги до розроблюваної системи.

У другому розділі даної магістерської дисертації було обґрунтовано вибір алгоритму кластеризації для модифікації та детально викладено рішення, що пропонується. Модифікація методу полягає в тому, що у порівнянні з базовим методом необхідно замінити вимірювання евклідової відстані, графовим вимірюванням, а також замінити центроїд на медіану набору графів.

У третьому розділі було проведено аналіз засобів розробки програмного забезпечення та описано розроблені системні модулі. А саме, класи, що відповідають за логіку додатку, які виконують підготовку даних до аналізу, кластеризацію та підготовку даних до візуалізації, а також відповідають за доступ системи до бази даних; контролери, які відповідають за передачу даних між клієнтом та сервером; компонент відображення користувацького інтерфейсу. Завдяки взаємодії цих компонентів в результаті було отримано такий веб-додаток, який може працювати у різноманітних браузерах, які надають операційні системи, без втрати сумісності з веб-середовищем.

У четвертому розділі даної дисертації було описано методологію тестування для розроблюваної системи, проаналізовано результати роботи та визначено напрями для подальшого вдосконалення системи у майбутньому. За результати тестування можна заявити, що розроблена система не має помилок, які можуть призвести до виключних ситуацій при роботі користувача.

За результатами проведеної роботи в останньому розділі було сформовано та побудовано бізнес-модель кінцевого продукту, що описує ключові моменти в організації діяльності, пов’язаної з поширенням розробленої системи для аналізу семантичного ядра на основі кластеризації пошукових запитів користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Guowang Miao, Jens Zander, Ki Won Sung, and Ben Slimane, Fundamentals of Mobile Data Networks, Cambridge University Press, ISBN 1107143217, 2016.
2. Perkins, C.E. Ad hoc On-demand Distance Vector (AODV) Routing / C.E. Perkins, C.E. Beldong-Royer // RFC 3561. – July 2003.
3. Broch, J.A performance comparation of multihop wireless ad hoc network routing protocols / J. Brocj, D.A. Maltz // Proc. Of MOBICOM`98 -1998.
4. Basagni, S. Mobility – Adaptive Protocols for Managing Large Ad Hoc Networks / S. Basagni. D. Turgut // Proceedings of the IEEE International Conference on Communication (ICC) – 2001 – p. 1539-1543.
5. Kawadia, V. System services for implementation Ad-Hoc routing: Architecture. Implementation and Experiences / V. Kawadia, Y Zhang, B, Gupta // Proceeding of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys) San Drancisko, C.A. – June 2010 – P. 99-112.
6. RFC 7541 — HPACK: Header Compression. Режим доступу https://tools.ietf.org/html/rfc7541
7. An Extensive Examination of Data Structures. Режим доступу: https://msdn.microsoft.com/en-us/library/ms379574(v=vs.80).aspx, вільний
8. Modeling mobility for vehicular ad-hoc networks Режим доступу: https://dl.acm.org/citation.cfm?id=1023892
9. Akyildiz I. F., Wang X., Wang W. Wireless mesh networks: a survey //Computer networks. – 2005. – Т. 47. – №. 4. – С. 445-487.
10. Perkins C. E. et al. Performance comparison of two on-demand routing protocols for ad hoc networks //IEEE Personal communications. – 2001. – Т. 8. – №. 1. – С. 16-28.

11. Ko Y. B., Vaidya N. H. Location□Aided Routing (LAR) in mobile ad hoc networks //Wireless networks. – 2000. – Т. 6. – №. 4. – С. 307-321.
12. Ko Y. B., Vaidya N. H. Location□Aided Routing (LAR) in mobile ad hoc networks //Wireless networks. – 2000. – Т. 6. – №. 4. – С. 307-321.
13. Daly E. M., Haahr M. Social network analysis for routing in disconnected delay-tolerant manets //Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing. – ACM, 2007. – С. 32-40.
14. Karp B., Kung H. T. GPSR: Greedy perimeter stateless routing for wireless networks //Proceedings of the 6th annual international conference on Mobile computing and networking. – ACM, 2000. – С. 243-254.
15. Intanagonwiwat C., Govindan R., Estrin D. Directed diffusion: A scalable and robust communication paradigm for sensor networks //Proceedings of the 6th annual international conference on Mobile computing and networking. – ACM, 2000. – С. 56-67.
16. Trifunovic S. et al. WiFi-Opp: ad-hoc-less opportunistic networking //Proceedings of the 6th ACM workshop on Challenged networks. – ACM, 2011. – С. 37-42.
17. Conti M. et al. Experimenting opportunistic networks with WiFi Direct //Wireless Days (WD), 2013 IFIP. – IEEE, 2013. – С. 1-6.
18. Arnaboldi V., Conti M., Delmastro F. CAMEO: A novel context-aware middleware for opportunistic mobile social networks //Pervasive and Mobile Computing. – 2014. – Т. 11. – С. 148-167.
19. Casetti C. et al. Content-centric routing in Wi-Fi direct multi-group networks //World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a. – IEEE, 2015. – С. 1-9.
20. Broch J. et al. A performance comparison of multi-hop wireless ad hoc network routing protocols //Proceedings of the 4th annual ACM/IEEE

international conference on Mobile computing and networking. – ACM, 1998. – С. 85-97.

21. Perkins C., Belding-Royer E., Das S. Ad hoc on-demand distance vector (AODV) routing. – 2003. – №. RFC 3561.
22. Lee S. J., Gerla M. AODV-BR: Backup routing in ad hoc networks //Wireless Communications and Networking Confernce, 2000. WCNC. 2000 IEEE. – IEEE, 2000. – Т. 3. – С. 1311-1316.
23. Chakeres I. D., Klein-Berndt L. AODVjr, AODV simplified //ACM SIGMOBILE Mobile Computing and Communications Review. – 2002. – Т. 6. – №. 3. – С. 100-101.
24. Eisenman, B. Learning React Native [Text] / Bonnie Eisenman. — 2nd edition. — Sebastopol: O'Reilly Media, 2017. — 242 p.
25. Kirupa, Understanding WebViews [Електронний ресурс] / Kirupa. — Режим доступу: https://www.kirupa.com/apps/webview.htm. — Дата доступу: 3.12.2019.
26. WebKit Org., The WebKit Opensource Project [Електронний ресурс] / WebKit Org. — Режим доступу: https://webkit.org/project/. — Дата доступу: 3.12.2019.
27. Garbade, M. Native vs. cross-platform app development: pros and cons [Електронний ресурс] / Dr. Michael J. Garbade. — Режим доступу: https://codeburst.io/native-vs-cross-platform-app-development-pros-and-cons-49f397bb38ac. — Дата доступу: 3.11.2019.
28. LeRoux, B. PhoneGap Beliefs, Goals, and Philosophy [Електронний ресурс] / Brian LeRoux. — Режим доступу: https://phonegap.com/blog/2012/05/09/phonegap-beliefs-goals-and-philosophy/. — Дата доступу: 2.11.2019.
29. Trice, A. PhoneGap Explained Visually [Електронний ресурс] / Andrew Trice. — Режим доступу: https://phonegap.com/blog/2012/05/02/phonegap-explained-visually/. — Дата доступу: 2.11.2019.

30. Wargo, J. Apache Cordova 4 Programming (Mobile Programming) [Text] / John M. Wargo. — 1st edition. — Boston: Addison-Wesley Profeccional, 2015. — 560 p.
31. Griffith, C. Mobile App Development with Ionic, Revised Edition: Cross-Platform Apps with Ionic, Angular, and Cordova [Text] / Chris Griffith. — 1st edition. — Sebastopol: O'Reilly Media, 2017. — 292 p.
32. Ionic, Core Concepts [Електронний ресурс] / Ionic. — Режим доступу: https://ionicframework.com/docs/intro/concepts. — Дата доступу: 2.11.2019.
33. Flutter, Flutter for iOS developers [Електронний ресурс] / Flutter. — Режим доступу: https://flutter.dev/docs/get-started/flutter-for/ios-devs. — Дата доступу: 2.11.2019.
34. Napoli, M. Beginning Flutter: A Hands On Guide to App Development [Text] / Varco L. Napoli. — 1st edition. — Birmingham: Wrox, 2019. — 528 p.
35. Aggarwal, R. 10 top Programming Languages in 2019 for Businesses [Електронний ресурс] / Ruchika Singh Aggarwal. — Режим доступу: https://codeburst.io/10-top-programming-languages-in-2019-for-developers-a2921798d652. — Дата доступу: 3.11.2019.
36. Google Inc., Building web apps in WebView [Електронний ресурс] / Google Inc. — Режим доступу: https://developer.android.com/guide/webapps/webview. — Дата доступу: 3.11.2019.
37. Kumar, S. How React Native Works? [Електронний ресурс] / Saket Kumar. — Режим доступу: https://www.codementor.io/saketkumar95/how-react-native-works-mhjo4k6f3. — Дата доступу: 3.11.2019.
38. Buckler, C. JavaScript HTML5 Programming [Text] / Craig Buckler. — 2nd edition. — Sebastopol: O'Reilly Media, 2012. — 241 p.

39. Morley, M. JSON-RPC 2.0 Specification [Електронний ресурс] / Matt Morley. — Режим доступу: https://www.jsonrpc.org/specification. — Дата доступу: 17.11.2019.
40. Turskyi, V. MOLE-RPC [Електронний ресурс] / Viktor Turskyi. — Режим доступу: https://www.npmjs.com/package/mole-rpc. — Дата доступу: 20.11.2019.
41. Facebook, Communication between native and React Native [Електронний ресурс] / Facebook. — Режим доступу: https://facebook.github.io/react-native/docs/communication-ios. — Дата доступу: 22.11.2019.
42. Vepsäläinen, J. SuviveJS – Webpack and React: From apprentice to master [Text] / Juho Vepsäläinen. — 2st edition. — Scotts Valley: CreateSpace Independent Publishing Platform, 2016. — 284 p.
43. Turskyi, V. Yet another JSON RPC Library? [Електронний ресурс] / Viktor Turskyi. — Режим доступу: https://docs.google.com/presentation/d/1NCmVJalBJp0Gliyc8KCdExHTl-xztDz3v50V4Y2k0bQ/edit#slide=id.g43ac3735fc_0_757. — Дата доступу: 22.11.2019.
44. Facebook, Out-of-Tree Platforms [Електронний ресурс] / Facebook. — Режим доступу: https://facebook.github.io/react-native/docs/out-of-tree-platforms. — Дата доступу: 30.11.2019.
45. Souders, S. Even Faster Web Sites: Performance Best Practices for Web Developers [Text] / Steve Souders. — 1st edition. — Sebastopol: O'Reilly Media, 2009. — 256 p.
46. Google Inc., About Android App Bundles [Електронний ресурс] / Google Inc. — Режим доступу: https://developer.android.com/guide/app-bundle. — Дата доступу: 30.11.2019.

47. ViroMedia, Overview [Електронний ресурс] / ViroMedia. — Режим доступу: https://docs.viromedia.com/docs/viro-platform-overview. — Дата доступу: 30.11.2019.
48. Gasston, P. The Modern Web: Multi-Device Web Development with HTML5, CSS3, and JavaScript [Text] / Peter Gasston. — 1nd edition. — San Francisco: No Starch Press, 2013. — 264 p.
49. Simpson, K. You Don’t Know JS: Async & Performance [Text] / Kyle Simpson. — 1nd edition. — Sebastopol: O’Reilly Media, 2015. — 296 p.
50. Simpson, K. You Don’t Know JS: Up & Going [Text] / Kyle Simpson. — 1nd edition. — Sebastopol: O’Reilly Media, 2015. — 88 p.

ДОДАТКИ

Додаток 1

Лістинг програми

Лістинг 1. kmeans_modify

```
import random
from collections import defaultdict
import numpy as np
from scipy.sparse import csr_matrix
class K_Means:

def __init__(self, k: int, iterations: int, data: csr_matrix, data_length:
int):
self.k = k
self.iterations = iterations
self.data = data
self.data_length = data_length
def __initialize_centroids(self):
centroids = []
centroid_norms = []
for random_index in random.sample(range(0, self.data_length - 1), self.k):
centroid = self.data.getrow(random_index).toarray()[0]
centroids.append(centroid)
return centroid_norms, centroids
def cluster(self, document_labels):
centroid_norms, centroids = self.__initialize_centroids()
clusters = defaultdict(set)
for i in range(self.iterations):
clusters = defaultdict(set)  # Reset the clusters
for row_index, sparse_row in enumerate(self.data):
vector = sparse_row.toarray()
closest_centroid = (float('inf'), None)
for centroid_index, centroid in enumerate(centroids):
dist = 1 - np.true_divide(np.dot(vector, centroid),
np.multiply(np.linalg.norm(vector), np.linalg.norm(centroid)))
if dist <= closest_centroid[0]:
closest_centroid = (dist, centroid_index)
clusters[closest_centroid[1]].add(row_index)
for centroid_index, vector_indices in clusters.items():
avg_vector = None
for vector_index in vector_indices:
if avg_vector is None:
avg_vector = self.data.getrow(vector_index).toarray()
else:
avg_vector = np.add(avg_vector, self.data.getrow(vector_index).toarray())
centroids[centroid_index] = (avg_vector / len(vector_indices))[0]
majority_sum = 0
for cluster in clusters.values():
labeled_document_counts = defaultdict(int)
for document_index in cluster:
labeled_document_counts[document_labels[document_index]] += 1
majority_class = (0, None)
for label, count in labeled_document_counts.items():
if count > majority_class[0]:
majority_class = (count, label)
majority_sum += majority_class[0]  # Add majority to global sum
print('Purity at iteration {} is\t{}'.format(i,
(majority_sum/self.data_length)))
return clusters
```

Лістинг 2. clusterisation.py

```
import re
import os
import csv
import argparse as cmdline
import gensim
import numpy as np
import sys
from numpy import *
import operator
import shutil

class Clustering():
def __init__(self, embedding_file_path, output_dir_path, threshold,
                  representative_vec, overlap_cluster, word_vector_dim):

self.embedding_file_path = embedding_file_path
self.output_dir_path     = output_dir_path
self.threshold                 = threshold
self.representative_vec  = representative_vec
self.overlap_cluster     = overlap_cluster
self.word_vector_dim     = word_vector_dim

def find_sentence_similarity(self, sentence1, sentence2):
sentence1_vector = np.zeros(int(self.word_vector_dim))
count = 0
index = 1

while count < int(self.word_vector_dim):
sentence1_vector[count] = float(sentence1[index])
index +=1
count +=1

sentence2_vector = np.zeros(int(self.word_vector_dim))
count = 0
index = 1
while count < int(self.word_vector_dim):
sentence2_vector[count] = float(sentence2[index])
index +=1
count +=1

cosine_distance = (np.dot(sentence1_vector,sentence2_vector)\
                                /(np.linalg.norm(sentence1_vector) * \
                                      np.linalg.norm(sentence2_vector)))
return cosine_distance

def create_new_cluster(self, number_of_clusters, sentence):
number_of_clusters = number_of_clusters + 1
filename = self.output_dir_path + '/cluster_' + str(number_of_clusters) +
'.csv'
self.write_to_csv_file(filename, sentence)
return number_of_clusters

def write_to_csv_file(self, filename, row):
if os.path.exists(filename):
output_file = open(filename,'ab+')
else:
output_file = open(filename,'wb+')
csv_writer  = csv.writer(output_file)
csv_writer.writerow(row)
output_file.close()
```

```
def cluster_sentences(self):
number_of_clusters = 0
with open(self.embedding_file_path,'r') as csvinput:
reader = csv.reader(csvinput)
# Create Output Directory if does not exist
if os.path.exists(self.output_dir_path):
shutil.rmtree(self.output_dir_path)
os.mkdir(self.output_dir_path)
# Read Embedding Input CSV File line by line
for row in reader:
if (number_of_clusters == 0):
number_of_clusters = self.create_new_cluster(number_of_clusters, row)
else:
matched_cluster_dict = {}
for subdir, dirs, cluster_files in os.walk(self.output_dir_path):
for cluster_file in cluster_files:
if 'rep_' in cluster_file:
cluster_fh = open(self.output_dir_path + '/' + cluster_file,'r')
csv_reader = csv.reader(cluster_fh)
c_row          = next(csv_reader)
similarity_score = self.find_sentence_similarity(row,c_row)

if (similarity_score > self.threshold):
matched_cluster_dict[(cluster_file.split('.')[0])[4:]] = similarity_score
cluster_fh.close()
def find_representative_sentence(self, number_of_clusters):
counter = number_of_clusters
while counter:
filename   = self.output_dir_path + '/cluster_' + str(counter) + '.csv'
read_file  = open(filename,'r')
csv_reader = csv.reader(read_file)
vec_sum    = np.zeros(self.word_vector_dim)
input_vec  = np.zeros(self.word_vector_dim)
num_rows = 0
for row in csv_reader:
num_rows += 1
column = 1
count  = 0
while count < int(self.word_vector_dim):
input_vec[count] = row[column]
column += 1
count  += 1
vec_sum = vec_sum + input_vec
read_file.close()

vec_avg = vec_sum/num_rows                cluster_rep_file  =
self.output_dir_path + '/rep_cluster_' + str(counter) + '.csv'
write_file = open(cluster_rep_file, 'wb+')
csv_writer = csv.writer(write_file)
row = ['Representative Vector']
if self.representative_vec == 'add':
row.extend(vec_sum)
else:
row.extend(vec_avg)
csv_writer.writerow(row)
write_file.close()
counter = counter - 1
```

Лістинг 3. data_preparer.py

```
import re
import csv
from collections import defaultdict
import numpy as np
from scipy.sparse import csr_matrix
class DataPreparer:
def __init__(self, file_path):
self.documents = []
self.classes = set()
self.document_labels = {}
self.document_count = 0
self.vocabulary_set = set()
self.vocabulary_size = 0
self.indexed_vocabulary = {}

self.word_document_count = defaultdict(int)
self.document_word_count = defaultdict(int)

self.__read_dataset(file_path)
def __read_dataset(self, file_path):
with open(file_path, 'r') as f:
for document in csv.reader(f):
self.classes.add(document[0])
self.document_labels[self.document_count] = document[0]
self.documents.append(" ".join(re.split(r'\W+', '{} {}'.format(document[1],
document[2]).lower())))
self.document_count += 1
def build_vocabulary(self):
# Count the words in all documents
for document in self.documents:
tmp_set = set()
for word in document.split():
if word not in tmp_set:
self.word_document_count[word] += 1
tmp_set.add(word)
# Find redundant words
redundant_words_set = set()
for key, value in self.word_document_count.items():
if value < 3 or value > self.document_count * 0.4 or key.isdigit():
redundant_words_set.add(key)
# Vocabulary properties
self.vocabulary_set = self.word_document_count.keys() - redundant_words_set
self.vocabulary_size = len(self.vocabulary_set)
# Build indexed vocabulary
vocabulary_dict = {}
for i, word in enumerate(sorted(list(self.vocabulary_set))):
vocabulary_dict[word] = i
self.indexed_vocabulary[i] = word
return vocabulary_dict
def generate_document_term_matrix(self):
vocabulary_dict = self.build_vocabulary()
sparse_matrix_dict = defaultdict(int)
for i, document in enumerate(self.documents):
for word in document.split():
if word in self.vocabulary_set:
sparse_matrix_dict[(i, vocabulary_dict[word])] += 1
self.document_word_count[i] += 1
return sparse_matrix_dict
```

Лістинг 4. main.py

```
import sys
import os
sys.path.insert(0, './word_embedding')
sys.path.insert(1, './clustering')
from word_embedding import WordEmbedding
from config_reader import ConfigParse
from cluster_sentences import Clustering

class MainExecutor():
def main(self):
# Create object of ConfigParser Class
config_obj = ConfigParse()

# Parse config file
print 'READING CONFIG FILE'
config_obj.config_reader()

# Create object of WordEmbedding Class
word_embedding_obj = WordEmbedding(config_obj.input_file_path,
                                   config_obj.word2vec_model,
                                   config_obj.word_vector_dim)

print 'CONVERTING INPUT SENTENCES TO VECTORS'
embedding_file = word_embedding_obj.sentence_to_vector()

# Create object of Clustering Class
clustering_obj = Clustering(embedding_file, config_obj.output_dir_path,
                            config_obj.threshold,
config_obj.representative_word_vector,
                            config_obj.cluster_overlap,
config_obj.word_vector_dim)

print 'CLUSTERING SENTENCES'
num_of_clusters = clustering_obj.cluster_sentences()
print str(num_of_clusters) + ' NUMBER OF CLUSTERS ARE GENERATED.'

# Remove Temporary Files
os.remove(embedding_file)
for subdir, dirs, cluster_files in os.walk(config_obj.output_dir_path):
for cluster_file in cluster_files:
if 'rep_' in cluster_file:
os.remove(config_obj.output_dir_path + '/' + cluster_file)

if __name__ == '__main__':
main_executor_obj = MainExecutor()
main_executor_obj.main()
```

Лістинг 5. main_page.html

```
<html class="s9-qk-short-lines" style="">
<head>
<meta charset="UTF-8"><script type="text/javascript">
//thx, h5bp.com
(function(a){
Function b(){}
for(var c="assert, count, debug, dir, dirxml, error, exception, group,
groupCollapsed, groupEnd, info, log, markTimeline, profile, profileEnd, time,
timeEnd, trace, warn".split(","),d;!!(d=c.pop());){
a[d]=a[d]||b;}})
(function(){
try{console.log();
return window.console;}
catch(a){return (window.console={});}}());
</script>
<link rel="x-s9-url" href="http://placeholder.protoshare.net" id="s9-
placeholder-service-url">
<meta name="x-s9-build-info" content="201907201528 / 201912150105 /
201912150105">
<link rel="icon" type="image/vnd.microsoft.icon"
href="//dg117jbyarqyn.cloudfront.net/9.5/7947/resources/favicon.ico">
<link media="all" rel="stylesheet" type="text/css" charset="UTF-8"
href="//dg117jbyarqyn.cloudfront.net/9.5/7947/resources/css.gz/viewer.css">
<link media="all" rel="stylesheet" type="text/css" charset="UTF-8"
href="/wa/styleSheet?oid=1" id="s9-custom-styles">
<script type="text/javascript"
src="//dg117jbyarqyn.cloudfront.net/lib/jquery_1.8.1/jquery.min.gz.js">
</script>
<script type="text/javascript"
src="//dg117jbyarqyn.cloudfront.net/lib/jquery.easing_1.3/jquery.easing.min.g
z.js">
</script>
<script type="text/javascript">jQuery.noConflict();</script>
<script type="text/javascript"
src="//dg117jbyarqyn.cloudfront.net/lib/swfobject_2.2/swfobject.gz.js">
</script>
<script type="text/javascript"
src="//dg117jbyarqyn.cloudfront.net/9.5/7947/resources/js.gz/viewer.js"></scr
ipt>
</head>
<body>
<script type="text/x-s9-marker" id="s9-layout-1000003"></script>
<div class="s9-ct s9--alive" style="position: absolute;
top:5px;left:5px;width:1040px;height:42px;">
<script type="text/x-s9-marker" id="s9-layoutsused-37"></script>
<div class="s9-content s9-scrolling" style="overflow: hidden; border-radius:
5px; background-color: rgb(44, 44, 44); box-shadow: none; width: 100%;
height: 100%;">
<div style="position: absolute; top:0px; left:865px; width:170px;
height:42px;" class="s9-st-value-btn s9--alive">
<script type="text/x-s9-marker" id="s9-layoutsused-34"></script>
<button type="button" disabled="disabled" value="0" class="s9-content"
style="color: rgb(255, 255, 255); font-size: 14px; font-family:
&amp;quot;Helvetica Neue&amp;quot;, Helvetica, Arial, sans-serif; border-color:
rgb(44, 44, 44); border-width: 0px; background-color: rgb(34, 34, 34); text-
align: right; white-space: pre-wrap; border-style: inset; box-shadow: rgba(0,
0, 0, 0.5) 0px 1px 3px inset; width: 100%; height: 100%;"><span
style="display: block;">Dasha Bilogub ▾</span>
</button>
</div>
```

```
<div class="s9-text s9--alive" style="position: absolute;
top:5px;left:11px;width:169px;">
<script type="text/x-s9-marker" id="s9-layoutsused-38"></script>
<div class="s9-content s9-scrolling" style="color: rgb(153, 153, 153); font-
size: 20px; font-family: &quot;Helvetica Neue&quot;, Helvetica, Arial, sans-
serif; white-space: pre-wrap; width: 100%;">
<div>Clusterisation tool</div>
</div>
</div>
<div class="s9-v-line s9--alive" style="position: absolute; top: 1px; left:
835px; height: 39px; border-color: rgb(61, 61, 61); border-style: groove;
border-width: 0px 0px 0px 2px;"><script type="text/x-s9-marker" id="s9-
layoutsused-36">
</script>
</div>
<div style="position: absolute; top:5px;left:890px;width:30px;height:30px;"
class="s9-img s9--alive"><script type="text/x-s9-marker" id="s9-layoutsused-
35">
</script>
<img src="/wa/asset?oid=1" class="s9-content" style="border-color: rgb(51,
51, 51); border-style: solid; border-width: 1px; background-color: rgb(255,
255, 255); width: 100%; height: 100%;">
</div>
</div>
</div>
<div style="position: absolute; top:130px;left:95px;" class="s9-img s9--
alive">
<script type="text/x-s9-marker" id="s9-layoutsused-55"></script>
<img alt="img1" src="/wa/asset?oid=2" class="s9-content">
</div>
<div style="position: absolute; top:150px; left:730px; width:215px
height:310px;" class="s9-text-area s9--alive">
<script type="text/x-s9-marker" id="s9-layoutsused-56"></script>
<textarea class="s9-content s9-scrolling" style="overflow: hidden; width:
100%; height: 100%;"></textarea>
</div>
<div class="s9-btn s9--alive" style="position: absolute;
top:520px;left:460px;width:171px;height:40px;">
<script type="text/x-s9-marker" id="s9-layoutsused-57"></script>
<button type="button" class="s9-content" style="color: rgb(255, 255, 255);
font-size: 14px; font-family: &quot;Helvetica Neue&quot;, Helvetica, Arial,
sans-serif; border-color: rgb(139, 231, 139); border-style: outset; border-
width: 1px; border-radius: 5px; padding: 5px; background-color: rgb(91, 183,
91); cursor: pointer; text-align: center; white-space: pre-wrap; width: 100%;
height: 100%;">
<span style="display: block;">Download file</span>
</button>
</div>
<script id="s9-info" type="text/x-s9-
info">{"layout":1000003,"layoutTypeTitle":"Design","project":1,"bodyStyle":nu
ll,"page":5}</script>

</body></html>
```

Лістинг 6. index.html

```
<!DOCTYPE html>
<html lang="en">
<!--  Maintenance Page Theme by Start Bootstrap and Jackie D'Elia Design -->
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Maintenance Page - Bootstrap Page</title>
    <!-- Bootstrap Core CSS -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="css/landing-page.css" rel="stylesheet">
    <!-- Custom Fonts -->
    <link href="font-awesome/css/font-awesome.min.css" rel="stylesheet"
type="text/css">
    <link
href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400i
talic,700italic" rel="stylesheet" type="text/css">
    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media
queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -
->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->
</head>
<body class="site">
    <div class="overlay">
        <!-- Header -->
        <div class="container">
            <div class="intro-message">
                <h1>Website Name</h1>
                <h2>Our website is currently down for maintenance.</h2>
                <hr class="intro-divider">
                <h3>In the meantime, you can find us on these social media
platforms.</h3>
                <ul class="intro-social-buttons">
                    <li>
                        <a href="#" class="btn btn-default btn-lg"><i
class="fa fa-facebook fa-fw"></i> <span class="network-
name">Facebook</span></a>
                    </li>
                    <li>
                        <a href="#" class="btn btn-default btn-lg"><i
class="fa fa-twitter fa-fw"></i> <span class="network-
name">Twitter</span></a>
                    </li>
                    <li>
                        <a href="#" class="btn btn-default btn-lg"><i
class="fa fa-pinterest fa-fw"></i> <span class="network-
name">Pinterest</span></a>
                    </li>
                </ul>
                <div class="intro-content">
                    <a class="tel" href="tel:+15555555555">555-555-5555</a>
                    <p class="address">123 Main St.
```

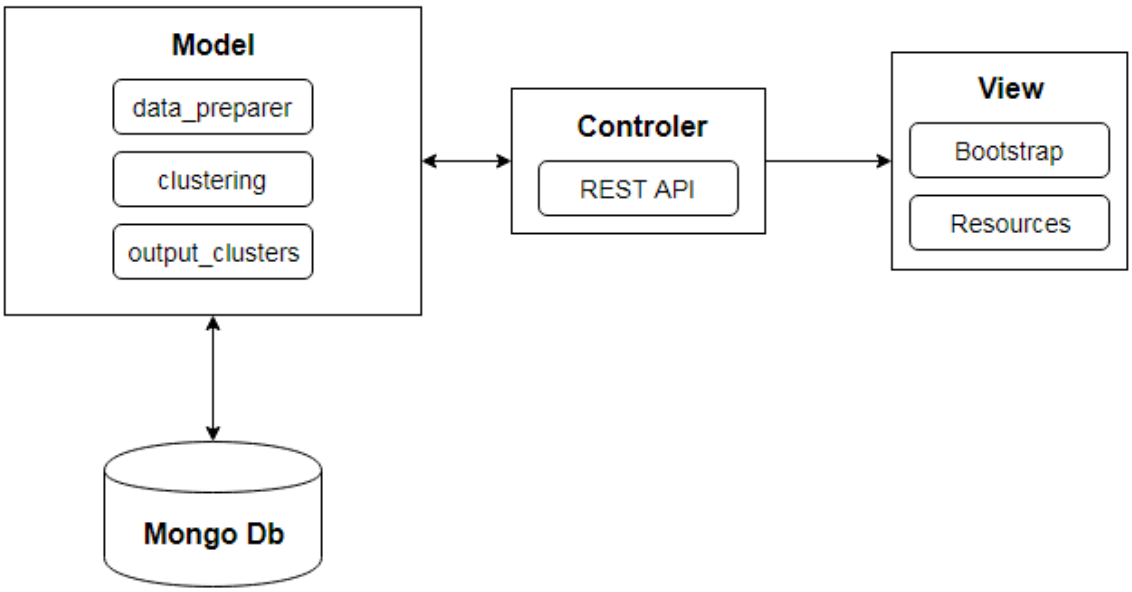
```
                        <br>Anytown, AB 55555
                    </p>
                </div>
            </div>
            <div class="intro-footer">
                <p class="copyright text-muted small">Copyright &copy; 2016.
All Rights Reserved</p>
            </div>
        </div>
        <!-- /.container -->
    </div>
    <!-- overlay -->
    <!-- jQuery -->
    <script src="js/jquery.js"></script>
    <!-- Bootstrap Core JavaScript -->
    <script src="js/bootstrap.min.js"></script>

</body>

</html>
```

Додаток 2

Копії графічних матеріалів



data_preparer	
+ __init__()	void
+ __read_dataset()	void
+ build_vocabulary()	void
+ generate_document_term_matrix()	void
+ apply_tf_idf()	void
+ __dict_to_sparse_matrix()	void

clustering_algorithm	
+ __init__()	void
+ find_sentence_similarity()	function
+ create_new_cluster()	function
+ cluster_sentences()	function
+ find_representative_sentence()	void

output_clusters	
+ __init__()	void
+ prepare_data()	void
+ write_to_csv_file()	function
+ store data()	void

Clusterisation tool

Dasha Bilogub

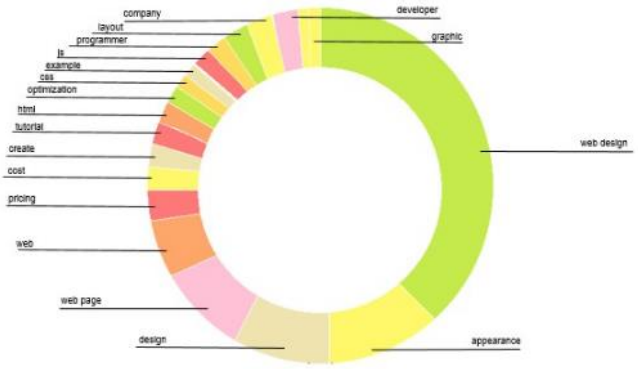
Enter semantic core of your site

Choose file or paste values

Выберите файл Файл не выбран

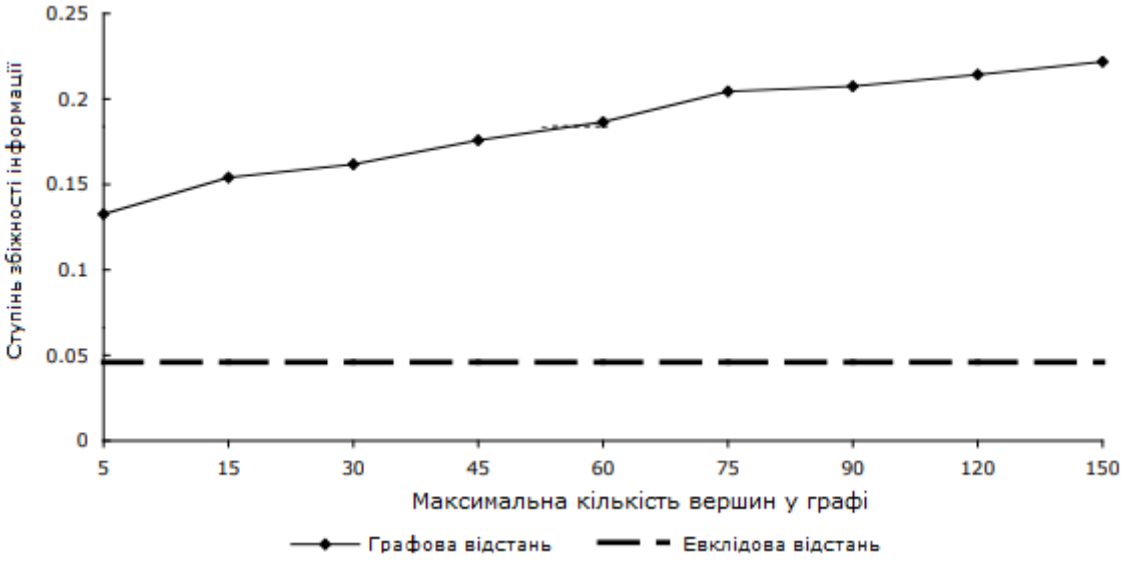
Clasterisation tool

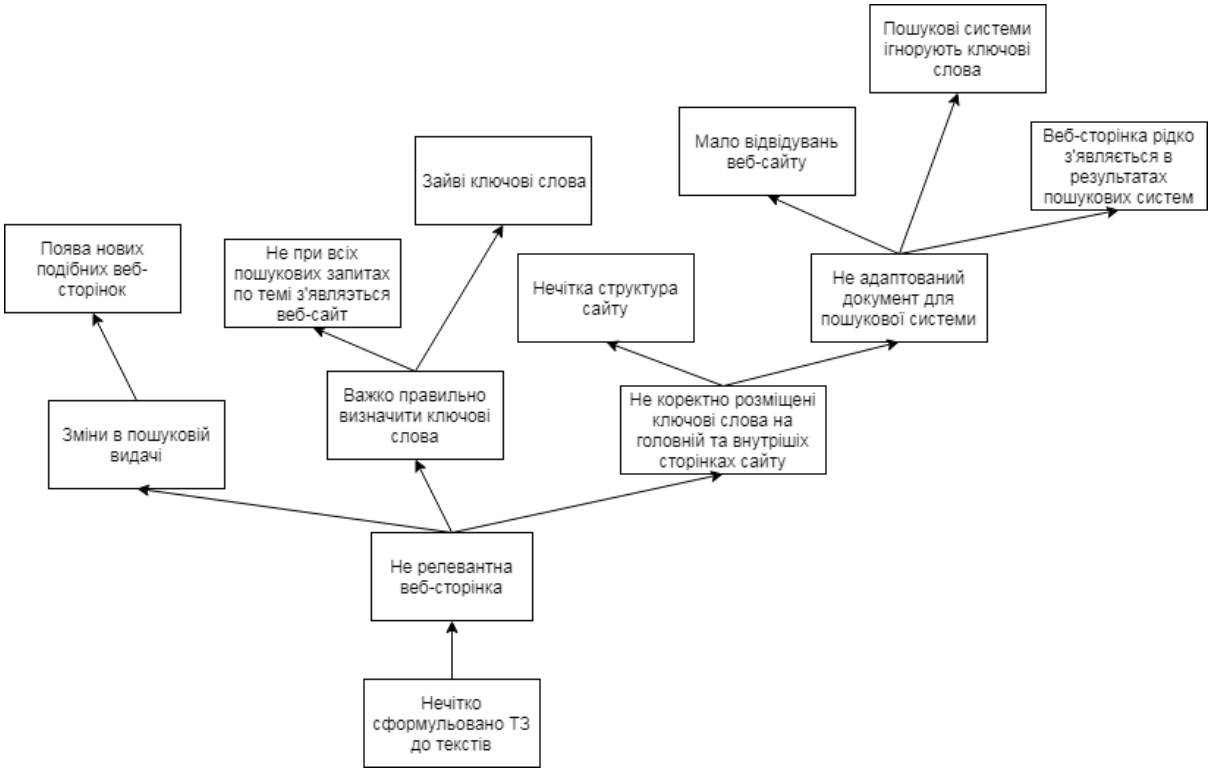
Dasha Bilogub



ecommerence web design
seo
seo and web marketing
professional logo design
graphic design studio
search engine marketing
agencie
developers
website building
website creation
search engine optimisation
graphic design studio
brand strategy
graphic design websites
web design
web design
web design philadelphia
web design company
brand definition
design website

Download file


0.25
0.2
0.15
0.1
0.05
0
Ступінь збіжності інформації
5
15
30
45
60
75
90
120
150
Максимальна кількість вершин у графі
Графова відстань
Евклідова відстань


Пошукові системи ігнорують ключові слова
Мало відвідувань веб-сайту
Веб-сторінка рідко з'являється в результатах пошукових систем
Зайві ключові слова
Поява нових подібних веб-сторінок
Не при всіх пошукових запитах по темі з'являэться веб-сайт
Нечітка структура сайту
Не адаптований документ для пошукової системи
Важко правильно визначити ключові слова
Не коректно розміщені ключові слова на головній та внутрішіх сторінках сайту
Зміни в пошуковій видачі
Не релевантна веб-сторінка
Нечітко сформульовано ТЗ до текстів

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНИЙ МЕТОД КЛАСТЕРИЗАЦІЇ ВЕБ-САЙТІВ НА ОСНОВІ АНАЛІЗУ ПОШУКОВИХ ЗАПИТІВ КОРИСТУВАЧІВ

Виконала: Білогуб Дар’я Сергіївна

Науковий керівник: к.т.н. Олещенко Любов Михайлівна

Київ – 2019

АКТУАЛЬНІСТЬ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

З огляду на постійну боротьбу пошукових систем, правильна структура сайту все більше виходить на перший план при проведенні пошукової оптимізації сайту. Один з основних ключів для грамотної опрацювання структури сайту – є максимально детальне опрацювання семантичного ядра. Працюючи з ядром, необхідно коректно визначити яка сторінка найточніше відповідає на конкретний пошуковий запит або групу запитів.

Об’єкт дослідження: процес аналізу ключових слів веб сторінок використовуючи інформацію про пошукові запити користувачів.

Предмет дослідження: моделі, методи, алгоритми кластерного аналізу пошукових запитів користувачів.

Мета дослідження: оптимізація релевантності документа запиту користувача шляхом розробки програмного забезпечення з використанням методів кластеризації пошукових запитів користувачів.

Наукове завдання: розробити метод кластеризації пошукових запитів користувачів.

ПОСТАНОВКА ЗАДАЧІ

1. Проаналізувати існуючі методів кластеризації.
2. Визначити основні переваги та недоліки існуючих методів для вирішення задачі кластеризації.
3. Розробити модифікований метод кластеризації на основі результатів виконаного аналізу.
4. Створити робочий проект програмного забезпечення для кластеризації пошукових запитів користувачів.
5. Створити стартап проект програмного забезпечення для кластеризації пошукових запитів користувачів.

АЛГОРИТМ K-MEANS

1. Початковий розподіл об'єктів по кластерам.
2. Ітеративний перерозподіл об'єктів по кластерам:

$$d(p,q) = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

3. Коли всі об'єкти розподілені по кластерам, заново перераховуються їх центри:

$$c_j = \frac{\sum_{i=1}^{L} x_i}{L}, \text{ де } x_i = C_j.$$

4. Якщо $c_j = c_{j-1}$, то це означає, що кластерні центри стабілізувалися і відповідно розподіл закінчено. Інакше переходимо до кроку 1.

АЛГОРИТМ K-MEANS

Переваги алгоритму k-середніх:

- простота використання;
- швидкість використання;
- зрозумілість і прозорість алгоритму.

Недоліки алгоритму k-середніх:

- алгоритм занадто чутливий до викидів, які можуть спотворювати середнє;
- результат залежить від вибору вихідних центрів кластерів, їх оптимальний вибір невідомий;
- число кластерів треба знати заздалегідь.

МЕТОДИ ТЕОРІЇ ГРАФІВ

Суть алгоритмів, що ґрунтуються на теорії графів, полягає в тому, що вибірка об'єктів представляється у вигляді графа:

$$G=(V,E),$$

де V – множина вершин (число вершин $|V|=n$), а E – множина ребер (число ребер $|E|=m$)

МЕТОДИ ТЕОРІЇ ГРАФІВ

Переваги графових алгоритмів кластеризації:

- наочність;
- відносна простота реалізації;
- можливість внесення різних удосконалень, заснованих на геометричних міркуваннях.

МАКСИМАЛЬНИЙ/МІНІМАЛЬНИЙ ЗАГАЛЬНИЙ ПІДГРАФ

Існує пряма залежність між вагою ребер графів та максимальним загальним підграфом між двома графами. Зокрема, вони є рівнозначними.

Припустимо, що:

$$g = mcs(G_1, G_2), \text{ якщо } g \subseteq G_1, g \subseteq G_2,$$
$$g = MCS(G_1, G_2), \text{ якщо } G_1 \subseteq g, G_2 \subseteq g$$

Загальний підхід полягає в тому, щоб створити граф сумісності для двох заданих графів, а потім знайти найбільше спільне в ньому.

МАКСИМАЛЬНИЙ/МІНІМАЛЬНИЙ ЗАГАЛЬНИЙ ПІДГРАФ

Функція з найнижчою вагою еквівалентна максимальному загальному підграфу між двома.

Для модифікації графу G_1 у G_2 потрібно лише виконати наступні дії:

1. Видалити вузли та ребра з G_1, які не відображаються у $mcs(G_1, G_2)$. Виконати будь-які підстановки вузлів або ребер.
2. Додати вузли та ребра з G_2, які не відображаються у $mcs(G_1, G_2)$.

МІРА ВІДСТАНЕЙ MCS

Визначено таку міру відстані:

$$d_{MCS}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{\max(|G_1|, |G_2|)}.$$

Ця міра відстані має чотири важливі властивості:

- обмеження у формуванні числа в інтервалі [0,1];
- відстань дорівнює 0 лише тоді, коли два графи однакові;
- відстань між двома графами симетрична;
- відстань задовольняє нерівність трикутника, що забезпечує вимірювання відстані в інтуїтивно зрозумілому вигляді.

МЕДІАНА НАБОРУ ГРАФІВ

Медіана набору графів S є графом g ($g \in S$) таким, що g має найменшу середню відстань до всіх елементів у S:

$$\mathrm{g} = \arg\min_{\forall s \in S}\left(\frac{1}{|S|}\sum_{i=1}^{|S|} d(s, G_i)\right)$$

Оскільки $g \in S$, то просто обчислити середню відстань до всіх графів для кожного графа в S. Медіана набору графів існує завжди.

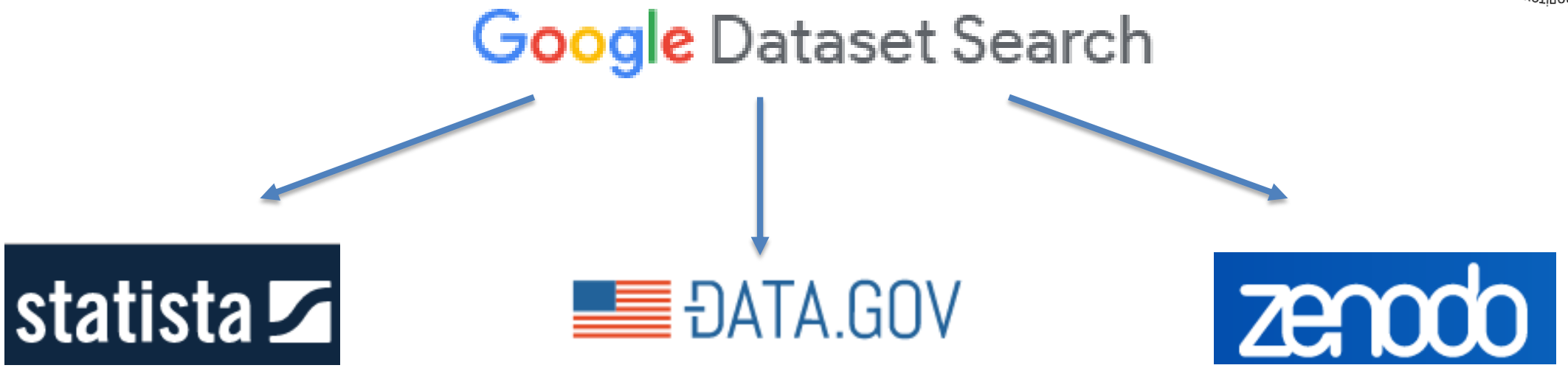
МОДИФІКАЦІЯ АЛГОРИТМУ K-MEANS

- замінити вимірювання відстані, використовуваного на кроці 3, графовим вимірюванням відстані;
- замінити центроїд, обчислений на етапі 2, медіаною набору графів.

МОДИФІКОВАНИЙ АЛГОРИТМ K-MEANS З ВИКОРИСТАННЯМ ТЕОРІЇ ГРАФІВ

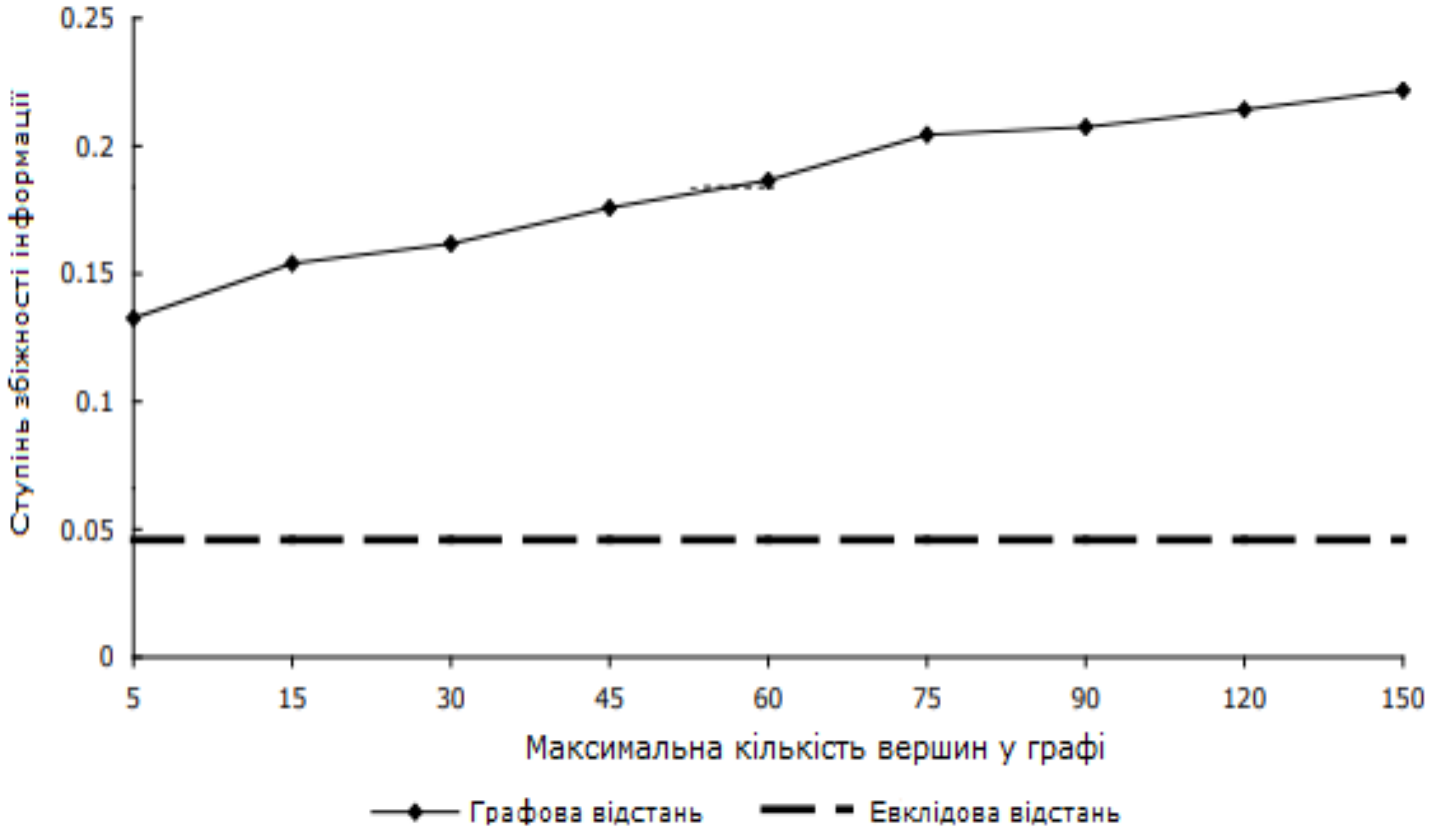
1. Призначити кожен елемент даних випадковим чином кластеру (від 1 до k).
2. Використовуючи початкове призначення, визначити медіану набору графів кожного кластеру.
3. З огляду на нові медіани, призначити кожен елемент даних в кластері його найближчої медіани, використовуючи графову міру відстані.
4. Повторно обчислити медіани, як на кроці 2. Повторити кроки 3 та 4, поки медіани не зміняться.

ТЕСТУВАННЯ МЕТОДУ



Search query	Topic Explorer Title	Topic Explorer Source
history of poverty in america	History of slavery	Wikipedia
women in the holocaust	Women in the military	Wikipedia
muslim terrorist in the united states	Islam in the United States	Wikipedia
the slave next door	The MIllionaire Next Door	Wikipedia
muslim schools in the us	Education in the United States	Gale Virtual Reference Library
islamic schools in the us	Education in the United States	Gale Virtual Reference Library
race in the media	Race and crime in the United States	Wikipedia
vawa what is it ?	Anarchy is What States Make of It	Wikipedia
domestic violence in the united states	Domestic partnership in the United States	Wikipedia
forced marriage united states	Desegregation busing in the United States	Wikipedia

РЕЗУЛЬТАТИ ТЕСТУВАННЯ МОДИФІКОВАНОГО МЕТОДУ

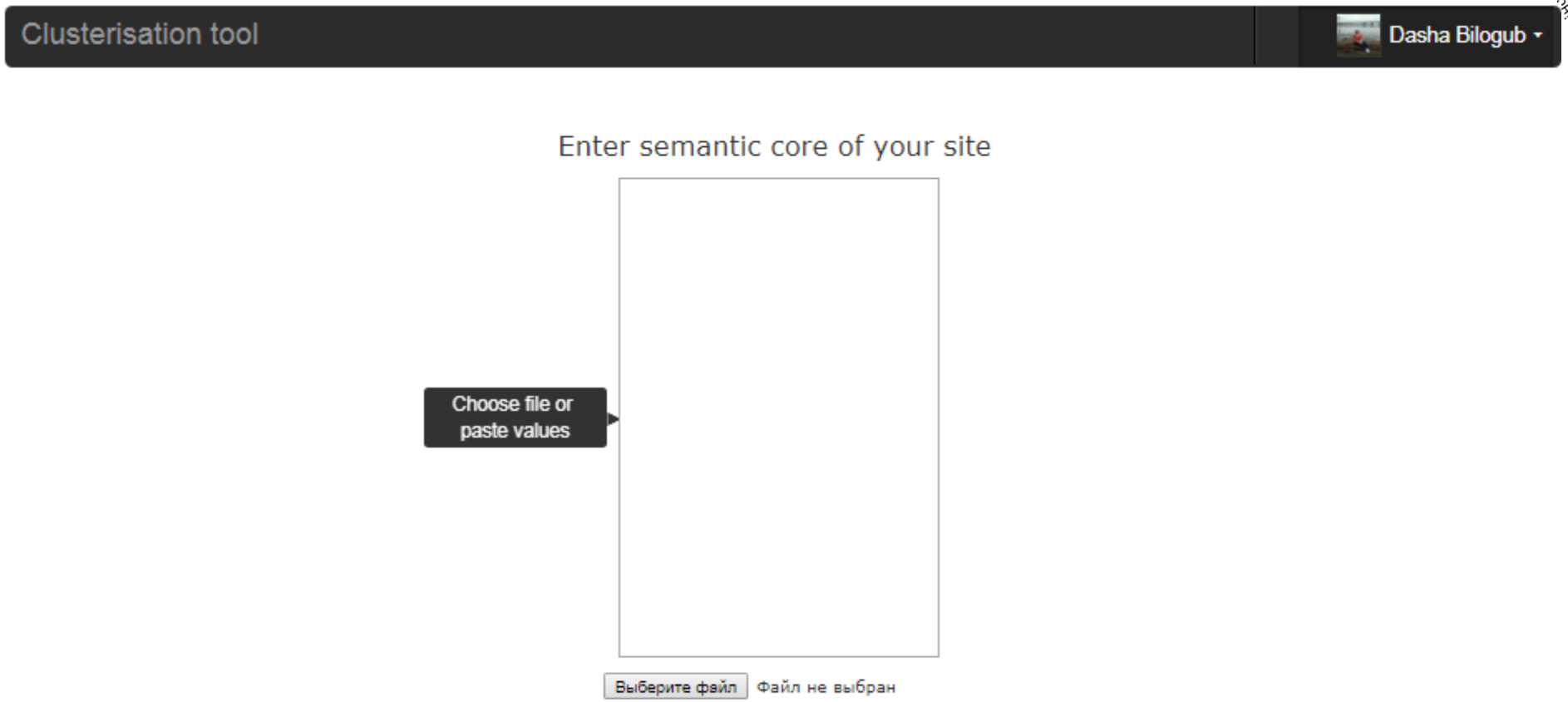


Діаграма порівняння результатів використання традиційного та модифікованого алгоритму за подібністю даних у кластері

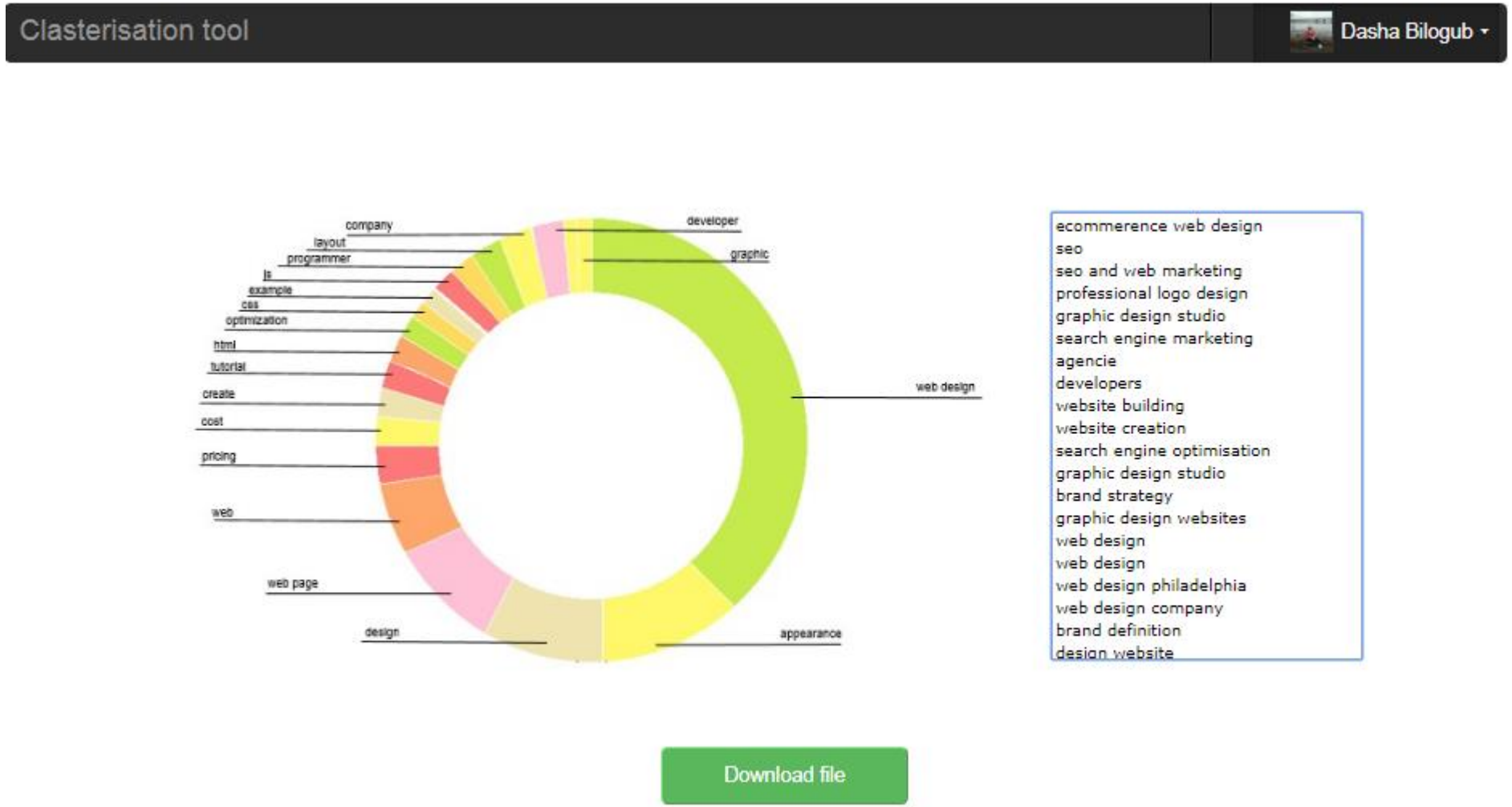
ВИКОРИСТАНІ ТЕХНОЛОГІЇ

Bootstrap

ПРИКЛАД ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КЛАСТЕРИЗАЦІЇ



ПРИКЛАД ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КЛАСТЕРИЗАЦІЇ ПОШУКОВИХ ЗАПИТІВ



БІЗНЕС-МОДЕЛЬ

<table>
<tr><th>Ключові партнери</th><th>Ключова діяльність</th><th>Ціннісна пропозиція</th><th>Відношення з клієнтами</th><th>Сегменти користувачів</th></tr>
<tr><td rowspan="3">1. Постачальники серверного та комп`ютерного обладнання.
2. Компанії, які використовують веб-застосунки.
3. Інвестори.</td><td>Розробка програмного забезпечення для підвищення релевантності веб-документу.</td><td rowspan="3">1. Програмне забезпечення, що реалізує кластеризацію запитів методом основаним на теорії графів.
2. Зручний інтерфейс.
3. Легкість використання.</td><td>1. Побудова лояльності</td><td rowspan="3">1. Компанії, які займаються розробкою сайтів.
2. Власники сайтів, SEO-менеджери</td></tr>
<tr><th>Ключові ресурси</th><th>Канали</th></tr>
<tr><td>1. Персонал.
2. Серверне обладнання.</td><td>1. Соціальні мережі.
2. Контекстна реклама.</td></tr>
<tr><th colspan="3">Структура витрат</th><th colspan="2">Джерела доходу</th></tr>
<tr><td colspan="3">1. утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат);
2. утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги);
3. податкові витрати;
4. оплата послуг юриста, бухгалтера, прибиральниці</td><td colspan="2">1. доходи від продажу ліцензій;
2. доходи від підтримки програмного забезпечення
3. підписки.</td></tr>
</table>

ФІНАНСОВИЙ ПЛАН СТАРТАПУ

ДОХОДИ (грн)	Довготривалі підписки	Усі функції системи	Обрані функції системи	Розширена технічна підтримка	Всього
Сумарно за рік:	147000	74000	54500	68000	401500

ВИТРАТИ (грн)	Технічна підтримка	Оплата праці	Оренда та комунальні витрати	Реклама	Всього
Сумарно за рік:	10500	155000	60000	47000	272500

ПРИБУТОК (грн) = 129 000

НАУКОВО-ІНОВАЦІЙНА НОВИЗНА

Запропонований модифікований метод кластеризації дозволяє підвищити точність результату у 3 рази, шляхом збереження структурної інформації запитів, яка зазвичай відкидається при використанні типового підходу векторної моделі. Даний метод перевершує метод на основі векторних k-means, що використовує евклідову відстань для обчислення відстані між кластерами.

ВИСНОВКИ

1. Проаналізовано існуючі методи кластеризації. Виявлено основні переваги та недоліки існуючих методів для вирішення задачі кластеризації пошукових запитів користувачів.
2. На основі виділених недоліків існуючих методів запропоновано модифікований метод кластеризації, який дозволяє зберігати структурну інформацію запитів, яка зазвичай відкидається при використанні типового підходу векторної моделі.

ВИСНОВКИ

3. Розроблено програмне забезпечення для кластеризації пошукових запитів користувачів, яке дозволяє користувачеві аналізувати ключові слова сайту шляхом їх порівняння з кластерами його пошукових запитів у мережі Інтернет.
4. Здійснено оцінку ефективності запропонованого методу.
5. Розроблено бізнес модель програмного забезпечення для кластеризації пошукових запитів користувачів.

АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

XII наукова конференція магістрантів та аспірантів «Прикладна математика та комп’ютинг» (ПМК-2019).

УНІКАЛЬНІСТЬ

100% - 2% = 98%

Дякую за увагу!